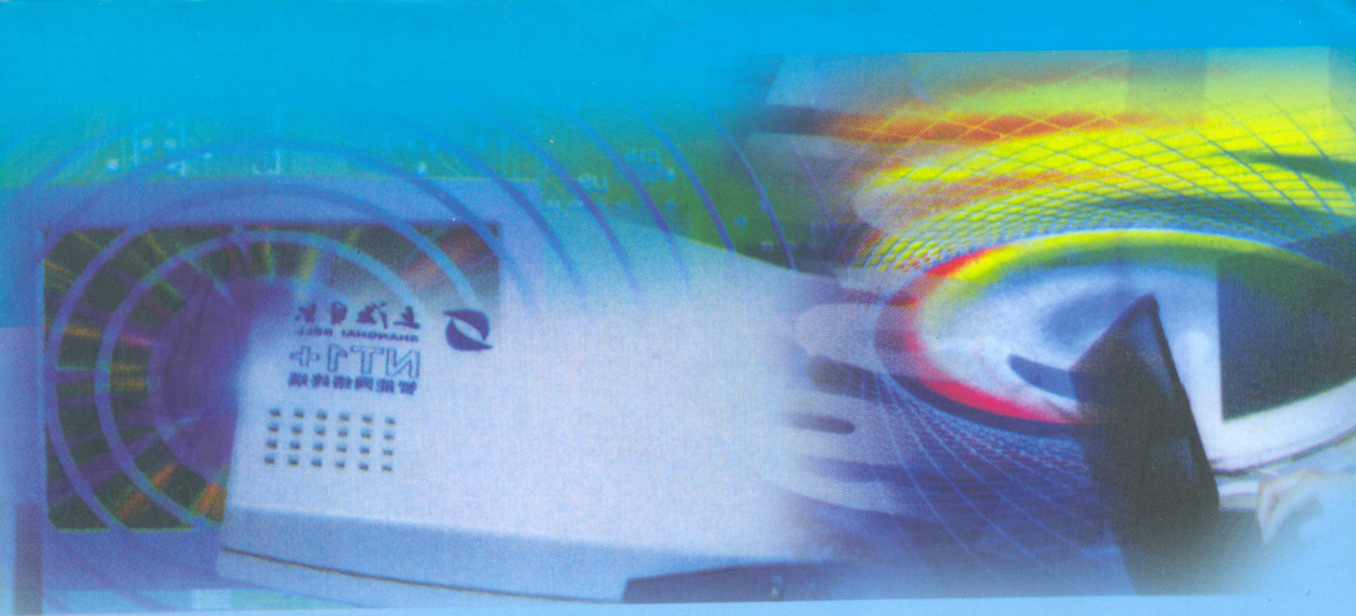


ĐỖ XUÂN TIẾN



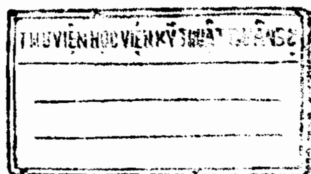
Kỹ thuật VI XỬ LÝ và lập trình ASSEMBLY CHO HỆ VI XỬ LÝ

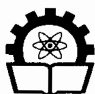


NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

PGS, TS. ĐỖ XUÂN TIẾN

KỸ THUẬT VI XỬ LÝ
VÀ
LẬP TRÌNH ASSEMBLY CHO HỆ VI XỬ LÝ





NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
HÀ NỘI

MỞ ĐẦU

Trong những thập niên cuối của thế kỷ 20, kỹ thuật điện tử đã liên tục có những tiến bộ vượt bậc, đặc biệt là trong kỹ thuật chế tạo mạch vi điện tử. Sự ra đời và phát triển nhanh chóng của kỹ thuật vi điện tử mà đặc trưng là kỹ thuật vi xử lý đã tạo ra một bước ngoặt quan trọng trong sự phát triển của khoa học tính toán và xử lý thông tin, nó ảnh hưởng quyết định đến con đường "tin học hoá" xã hội, tức là con đường mà thông tin đã và đang trở thành lực lượng sản xuất trực tiếp trong nền sản xuất của kỷ nguyên này.

Các bộ vi xử lý ngày nay được xây dựng trên cơ sở của một tấm bán dẫn silic vi điện tử cỡ lớn (LSI) và cực lớn (VLSI) nên kích thước nhỏ gọn, tốc độ cao, tiêu hao năng lượng thấp và đặc biệt là giá rẻ nên chúng ngày càng phổ dụng trong các ứng dụng không chỉ trong các chuyên ngành đặc thù như kỹ thuật điện tử, tin học, viễn thông, công nghệ thông tin, kỹ thuật điều khiển tự động mà cả trong các lĩnh vực phi điện tử khác. Năm 1970 công ty trẻ tuổi Intel cho ra đời bộ vi xử lý đầu tiên, có tên gọi là Intel 4004, nhằm đáp ứng nhu cầu cấp thiết của một công ty kinh doanh là hãng truyền thông BUSICOM. Intel-4004 là kết quả của một ý tưởng quan trọng trong sự phát triển của kỹ thuật xử lý số. Đó là một kết cấu logíc (Automat hữu hạn) mà có thể thay đổi chức năng của nó bằng chương trình ngoài chứ không phát triển theo hướng tạo một cấu trúc cứng chỉ thực hiện theo một số chức năng nhất định như trước đây. Do đó khả năng mềm dẻo hoá trong các thao tác của mình mà Intel-4004, vào năm 1971 đã trở thành bộ vi xử lý đầu tiên của thị trường thế giới. Intel-4004 là bộ vi xử lý 4 bít song song, được chế tạo theo quy trình công nghệ MOS kênh cảm ứng loại P. Thời gian tối thiểu để thực hiện một lệnh là 10,8 μs. Năm 1972 hãng Intel cho xuất xưởng bộ vi xử lý có tên gọi là Intel-8008. Kiểu này vẫn được chế tạo theo công nghệ PMOS nhưng là loại 8 bit song song. Bộ vi xử lý này là CPU của máy vi tính MICRAL do Pháp chế tạo. Đến đây hàng loạt các hãng điện tử nổi tiếng hàng đầu của thế giới như hãng National, Rockwell, Fairchild, Mostek... đã nhanh chóng đi vào công nghệ sản xuất và chế tạo các bộ vi xử lý.

Năm 1974 hãng Intel cho ra đời bộ vi xử lý 8080- 8 bit song song được chế tạo theo công nghệ NMOS với thời gian để thực hiện một lệnh là 2 μs đã đánh dấu một bước tiến lớn trên con đường làm chủ tốc độ xử lý tin của kỹ thuật điện tử hiện đại. Các hãng khác cũng cho ra đời những bộ vi xử lý có tính năng tương

ứng: 6800 (Motorola), 8080(texas Instrument), TLCS12 (Toshiba), 8080 (NEC), F8 (Fairchild), 2650 (Signetics), IM6100 (Intersil), 2650 (RTC), 8080A,4040 (Sinmens), 2900 và và 6800 (của hãng Sescosem)...Năm 1978, loại 8080 được cải tiến thành loại 8085. Lúc này đã xuất hiện những máy tính mini sử dụng các bộ vi xử lý nói trên. Theo đà đó các thông số cơ bản của bộ vi xử lý ngày càng được cải thiện: tốc độ ngày càng lớn (các bộ vi xử lý hiện đại của INTEL đã đạt tới tốc độ nhiều GHZ), độ rộng kênh thông tin ngày càng lớn (các bộ vi xử lý hiện đại của INTEL có kênh dữ liệu 16/32/64 bit). Điều đó đã giúp cho bài toán thiết kế các hệ vi xử lý chuyên dụng với tính năng rộng lớn trở nên dễ dàng hơn.

Một hệ vi xử lý tối thiểu phải bao gồm một bộ vi xử lý (đó là khối điều khiển và xử lý trung tâm CPU), một bộ nhớ RAM, một bộ nhớ cố định ROM và các cổng vào ra số liệu cùng những thiết bị ngoại vi cần thiết. Một hệ vi xử lý tối đa không giới hạn về số lượng thành phần, về chức năng thực hiện và về quy mô ứng dụng. Vấn đề là trên cơ sở của yêu câu cụ thể của hệ cần thiết kế mà tổ chức được phần cứng của hệ ở dạng tối thiểu (nhằm tăng tốc độ, giảm giá thành và tăng độ tin cậy) và xây dựng phần mềm điều khiển thật tối ưu nhằm tăng khả năng linh hoạt và mềm dẻo trong các phép xử lý, gia công và biến đổi tín hiệu mà hệ phải thực hiện.

Những vấn đề chính đã nêu ở trên sẽ được khảo sát, nghiên cứu trong những chương sau của giáo trình này. Vì để cung cấp các kiến thức cơ bản cho các đối tượng là *sinh viên đại học của các chuyên ngành điện tử-viễn thông, công nghệ thông tin và tự động điều khiển* nên trong khuôn khổ của cuốn tài liệu không dài, tác giả đã cố gắng tổng hợp, cập nhật các tài liệu thiết yếu trong và ngoài nước để xây dựng nội dung tài liệu này nhằm đạt mục tiêu đã đề ra. Nội dung của tài liệu bao gồm 12 chương, 2 phụ lục. Cụ thể:

Chương 1. Kiến trúc hệ vi xử lý. Giới thiệu kiến trúc của hệ Vi xử lý tối thiểu. Tổ chức và đặc trưng của kênh hệ thống, chức năng bộ nhớ ROM, RAM và phương pháp quản lý bộ nhớ trung tâm của hệ vi xử lý.

Chương 2. Bộ vi xử lý 16 bit 8086 INTEL. Trình bày cấu trúc phần cứng và nguyên tắc làm việc của bộ vi xử lý 16/32 bit thông qua 80286. Các tín hiệu và chức nàng của chúng. Phương pháp quản lý bộ nhớ ở chế độ địa chỉ thực và chế độ địa chỉ ảo của bộ vi xử lý 16/32 bit.

Chương 3. Lập trình assembly cho hệ vi xử lý. Trình bày tổng quan về ngôn ngữ assembly và các thành phần cơ bản của nó. Trình bầy bộ ký tự, từ khoá, cú pháp câu lệnh, các lệnh giả, các toán tử cùng trình biên dịch MACRO ASSEMBLER. Tiếp theo là giới thiệu tập lệnh của bộ vi xử lý 80286 cùng với cách phân chia chúng theo nhóm để tiện cho khảo sát.

Chương 4. Thiết kế hệ vi xử lý chuyên dụng. Trình bày trình tự và phương pháp thiết kế các hệ vi xử lý chuyên dụng theo chức năng yêu cầu. Minh hoạ bằng thiết kế hệ thu thập tín hiệu đa kênh.

Chương 5. Cổng trao đổi thông tin với ngoại vi. Trình bày phương pháp vào/ra thông tin tách biệt và phương pháp vào/ra thông tin theo địa chỉ bộ nhớ. Vi mạch ghép nối có lập trình 8255A và phương pháp ghép nối 8255A với hệ vi xử lý.

Chương 6. Chế độ ngắt của bộ vi xử lý. Trình bày chế độ ngắt của bộ vi xử lý. Tổ chức ngắt và nguyên tắc hoạt động của ngắt trong hệ vi xử lý 80X86. Chip điều khiển ngắt ưu tiên 8259A. Ghép nối Chip 8259A với hệ vi xử lý.

Chương 7. Truyền thông tin nối tiếp. Trình bày các khái niệm về truyền số liệu. Mạch thu phát dị bộ vạn năng IN8250A/16450. Mạch thu phát đồng bộ và dị bộ vạn năng USART 8251A. Nối ghép UART 8250A và USART 8251A với hệ vi xử lý.

Chương 8. Biến đổi tín hiệu tương tự-số và tín hiệu số - tương tự. Trình bày nguyên tắc hoạt động của bộ biến đổi số - tương tự và bộ biến đổi tương tự- số. Bộ biến đổi ADC 8 bit 0809. Bộ biến đổi ADC 12 bit AD574A. Ghép nối ADC 0809 và AD574A với hệ vi xử lý.

Chương 9. Hệ vi xử lý ON-CHIP. Trình bày cấu trúc của hệ vi xử lý On-chip 80C51 (và 89C51). Tổ chức cổng vào/ra của On-chip. Khối tạo thời gian và bộ đếm của On-chip. Cơ chế ngắt của On-chip 80C51.

Chương 10. Tập lệnh của hệ vi xử lý ON-CHIP 80C51. Trình bày cấu trúc lệnh và nguyên lý thực hiện lệnh của hệ vi xử lý on-chip 80C51. Trình bầy tập lệnh của On-chip 80C51 cùng các nhóm lệnh chuyển dữ liệu, nhóm lệnh điều khiển biến logic, nhóm lệnh rẽ nhánh chương trình, nhóm lệnh tính toán các phép tính số học và logic.

Chương 11. Thiết kế hệ xử lý trên hệ vi xử lý ON-CHIP 80C51. Trình bày trình tự thiết kế hệ vi xử lý chuyên dụng trên hệ vi xử lý on-chip. Minh hoạ bằng các thí dụ thiết kế hệ chức năng.

Chương 12. Hệ xử lý song song. Trình bày kiến trúc của hệ xử lý song song. Kiến trúc kiểu PIPELINE. Kiến trúc kiểu đa CPU. Lập trình cho hệ xử lý song song đa CPU. Minh hoạ trên thí dụ xử lý song song các tham số ảnh.

Trong các chương đều có các thí dụ minh hoạ, đặc biệt là các chương liên quan tới bài toán thiết kế hệ vi xử lý chuyên dụng. Trong các minh hoạ đó đã thể hiện một cách nhất quán các bước thực hiện thiết kế hệ thống từ khâu phân tích yêu cầu nhiệm vụ tới khâu tổ chức phần cứng và xây dựng phần mềm MONITOR tương ứng cho hệ cần thiết kế. Vấn đề ghép nối với máy tính cũng được đặt ra nhằm tận dụng khả năng mạnh của máy tính trong các bài toán xử lý cấp 2, cấp 3 cho các cấu trúc tin phức tạp.

Tác giả xin chân thành cảm ơn Pgs. Ts Đỗ Đức Giáo và Pgs. Ts Đỗ Trung Tuấn (Trường Đại học Khoa học Tự nhiên thuộc Đại học Quốc gia Hà Nội) đã bỏ nhiều công sức để hiệu đính cuốn tài liệu này, cảm ơn Nhà xuất bản Khoa học và Kỹ thuật đã tạo mọi điều kiện thuận lợi để cuốn sách này đến tay bạn đọc.

Do khả năng và thời gian chuẩn bị bản thảo còn hạn chế, cuốn sách chắc chắn không tránh được các thiếu sót, chúng tôi mong nhận được sự góp ý chân thành của các bạn đọc. Thư góp ý xin gửi về nhà xuất bản Khoa học và Kỹ thuật - 70 Trần Hưng Đạo Hà Nội.

Tác giả

Chương 1
KIẾN TRÚC HỆ VI XỬ LÝ

Khi nói tới hệ vi xử lý người ta dùng danh từ kiến trúc (Architecture) để chỉ tổ chức của hệ vì muốn nhấn mạnh đặc trưng cơ bản của hệ vi xử lý gồm hai mặt thống nhất không thể tách rời là phần mềm cài đặt trong hệ có nhiệm vụ vận hành chức năng hệ thống trên nền cấu trúc phần cứng của hệ.

1.1. TỔ CHỨC CHUNG CỦA HỆ VI XỬ LÝ

Tổ chức chung của hệ vi xử lý được thể hiện trên hình 1.1, bao gồm các thành phần chính như sau:

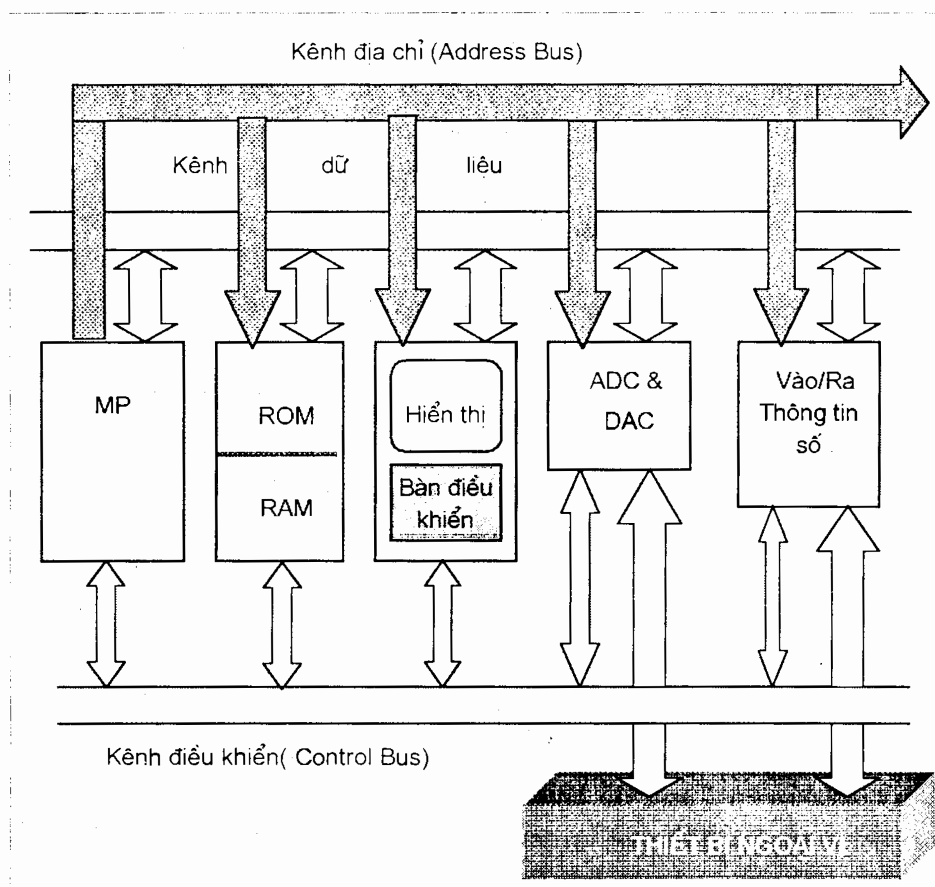
Bộ vi xử lý (VXL) là hạt nhân của hệ vi xử lý, nắm quyền kiểm soát toàn bộ các thành phần có trong hệ thống, thực hiện mọi thao tác liên quan tới các phép tính toán, xử lý, gia công, biến đổi các dạng tín hiệu nhị phân trong hệ dưới sự điều khiển của chương trình cài đặt trong bộ nhớ trung tâm của hệ vi xử lý. Bộ vi xử lý còn được gọi tên viết tắt là MP (Microprocessor - bộ vi xử lý), CPU (Central Processing Unit - khối xử lý trung tâm)

Bộ nhớ trung tâm gồm hai thành phần. Thành phần thứ nhất là bộ nhớ cố định ROM (Read Only Memory) dùng để chứa chương trình điều hành hoạt động của hệ vi xử lý, cho nên nó còn được gọi là chương trình MONITOR (người hướng dẫn). ROM còn dùng để chứa số liệu các bảng, biểu và các tham số hệ thống cũng như các số liệu cố định của hệ thống. Nội dung của ROM phải được chuẩn bị trước khi ghép nó vào hệ. Trong quá trình hoạt động sau này, nội dung đó không bị thay đổi. Thành phần thứ hai là bộ nhớ đọc/ghi RAM (Random Access Memory) dùng làm môi trường xử lý thông tin, để lưu trữ các kết quả trung gian và kết quả cuối cùng của các phép tính toán, xử lý thông tin. Nó cũng dùng để tổ chức các vùng đệm dữ liệu (Data Buffer) trong các thao tác thu, phát, chuyển đổi số liệu. Nói chung RAM là bộ nhớ dữ liệu động mà nội dung của nó có thể thay đổi trong quá trình hoạt động của hệ vi xử lý.

Bộ hiển thị dùng để thể hiện trạng thái hoạt động của hệ vi xử lý. Phụ thuộc vào nhiệm vụ và qui mô của hệ mà cấu trúc bộ hiển thị có thể từ rất đơn

giản đến rất phức tạp. Bàn điều khiển dùng để đưa lệnh, dữ liệu cần thiết vào hệ vi xử lý. Tương tự như bộ hiển thị, phụ thuộc vào nhiệm vụ và qui mô của hệ mà cấu trúc bàn điều khiển có thể từ rất đơn giản đến rất phức tạp.

Khối xuất nhập thông tin số dùng để trao đổi thông tin với các thiết bị ngoại vi làm việc theo nguyên tắc số. Đối với các thiết bị ngoại vi làm việc theo nguyên tắc phi số thì cần biến đổi dạng tín hiệu để hệ vi xử lý và thiết bị ngoại vi có thể hiểu được nhau. Các bộ biến đổi đó là bộ biến đổi tín hiệu liên tục sang tín hiệu số ADC (Analog To Digital Converter) và bộ biến đổi tín hiệu số sang tín hiệu liên tục DAC (Digital To Analog Converter).



Hình 1.1. **Kiến trúc chung của hệ Vi xử lý**

Kênh thông tin hệ thống bao gồm ba thành phần:

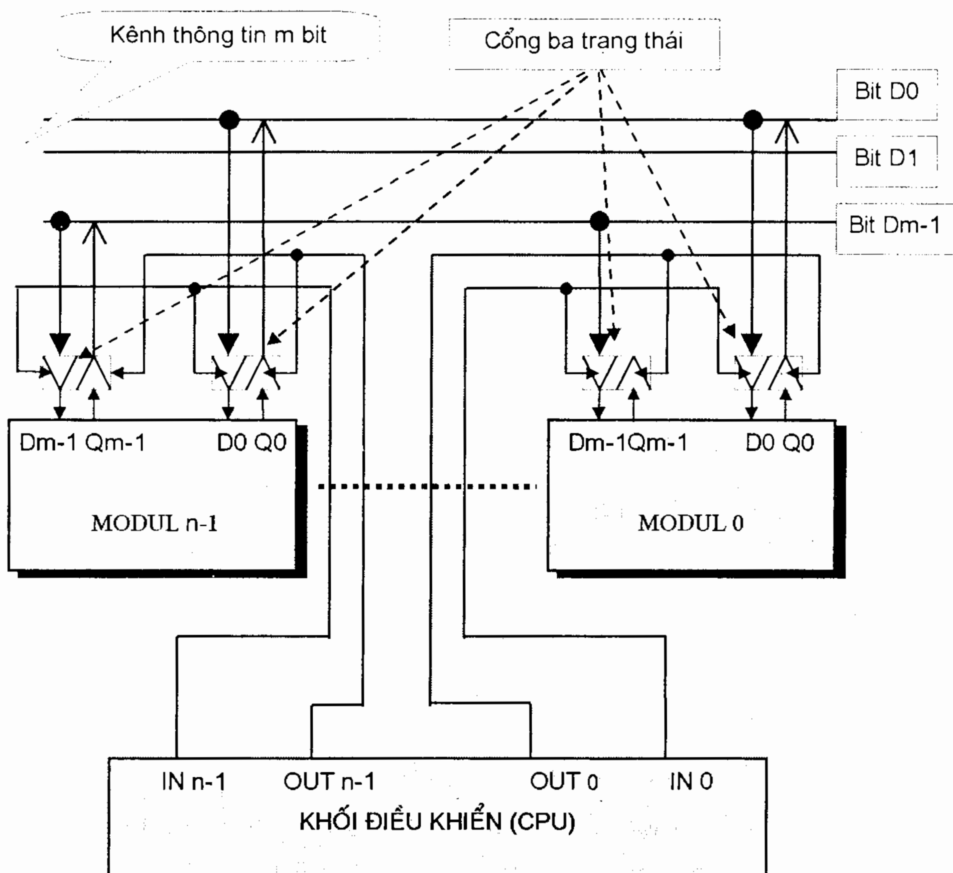
Thành phần thứ nhất là kênh địa chỉ (Address Bus). Đây là kênh một chiều đi từ bộ vi xử lý ra. Bộ vi xử lý sử dụng kênh này để quản lý các thành phần có trong hệ bằng cách gán cho mỗi thành phần một địa chỉ xác định.

Thành phần thứ hai là kênh dữ liệu (Data Bus). Đây là kênh hai chiều dùng để trao đổi thông tin giữa bộ vi xử lý và các thành phần có trong hệ.

Thành phần thứ ba là kênh điều khiển (Control Bus). Đây là tập hợp các dây tín hiệu điều khiển để tạo liên lạc giữa bộ vi xử lý và các thành phần có trong hệ nhằm đồng bộ hoá mọi chế độ và mọi thao tác của hệ thống.

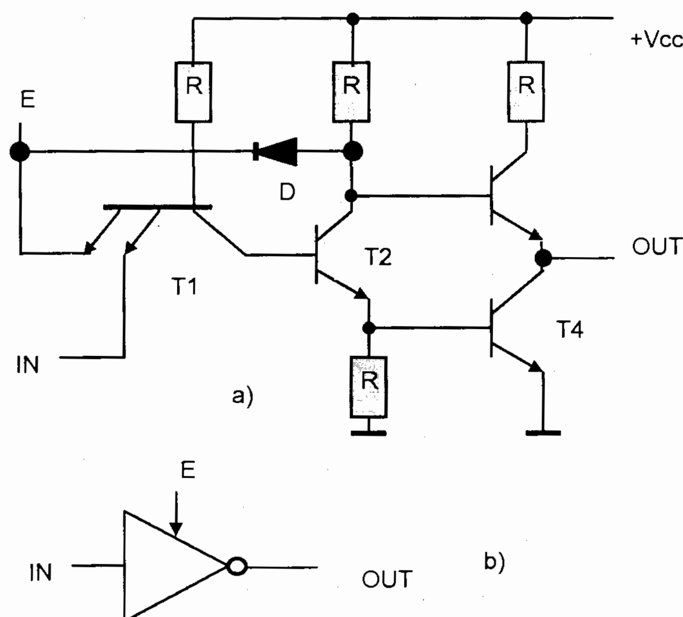
1.2. TỔ CHỨC KÊNH THÔNG TIN TRONG HỆ VI XỬ LÝ

Kênh thông tin hệ thống là kênh song song, tức là kênh dùng chung cho tất cả các thành phần có trong hệ. Mặt khác cấu trúc kênh là kênh kỹ thuật, tức là tồn tại các tham số đường dây như trở kháng đường dây, điện cảm, điện dung ký sinh nên nếu số lượng các thành phần trong hệ lớn thì sẽ xảy ra hiện tượng quá tải kênh thông tin. Hệ quả là mức lôgic 0 và mức lôgic 1 có thể bị nhầm lẫn. Để tránh xảy ra trường hợp không mong muốn ở trên, cần tổ chức cổng kiểm soát trạng thái hoạt động cho mỗi thành phần (hình 1.2). Cổng này có nhiệm vụ tạo ra trạng thái đặc biệt khi thành phần không được kích hoạt làm việc. Trạng thái đặc biệt này sẽ cách ly về mặt tín hiệu giữa kênh với thành phần dù rằng các dây tín hiệu vẫn được nối với nhau về mặt vật lý.



Hình 1.2. **Tổ chức kênh thông tin trong hệ Vi xử lý**

Cấu kiện điện tử thực hiện chức năng của cổng kiểm soát này được biểu diễn trên hình 1.3a. Đây là mạch logic TTL của hàm NOT. Khi chân E có mức điện áp cao thì diode D bị thiên áp ngược nên mạch logic này hoạt động bình thường: hoặc nó ở trạng thái logic 0 hoặc nó ở trạng thái logic 1. Khi chân E có mức điện áp thấp thì diode D đươc thiên áp thuận nên qua diode này sẽ có dòng chảy qua làm cả hai Transistor ở lối ra bị ngắt nên điểm OUT và điểm IN của mạch logic bị cách ly hoàn toàn. Nói cách khác, mạch logic này được đặt ở trạng thái đặc biệt, trạng thái trở kháng cao. Những cổng kiểm soát như thế được gọi là mạch ba trạng thái (hình 1.3 b). Hai trạng thái đầu là trạng thái logic 0 và trạng thái logic 1, còn trạng thái thứ ba là trạng thái trở kháng cao.

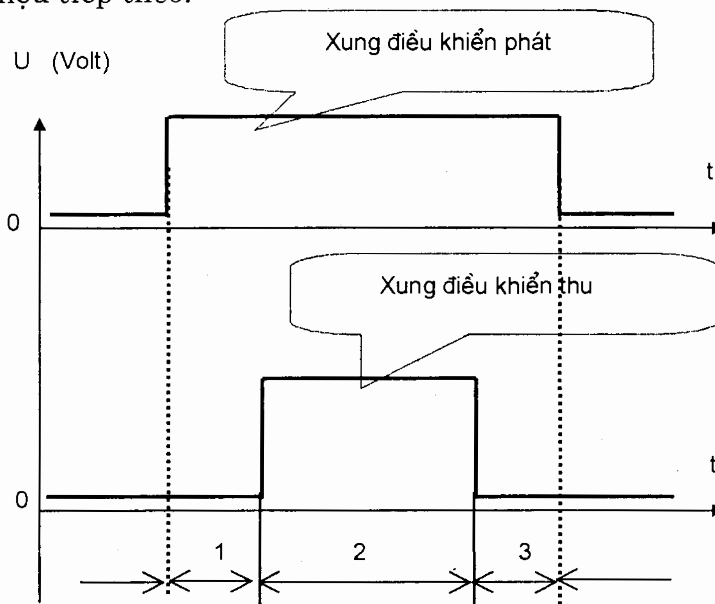


Hình 1.3. **Mạch 3 trạng thái (a), ký hiệu (b)**

Bộ vi xử lý sẽ điều khiển hoạt động của các cổng ba trạng thái này. Do bộ vi xử lý tại từng thời điểm chỉ có thể làm việc được với một thành phần có trong hệ nên nó chỉ đặt cổng ba trạng thái của thành phần tương ứng lên trạng thái làm việc, còn các thành phần khác bị treo lên trạng thái trở kháng cao. Điều đó có nghĩa nếu được thiết kế đúng thì kênh thông tin chỉ chịu tải của hai thành phần tại thời điểm đang xét mà thôi (thành phần xuất và thành phần nhập thông tin) nên kênh không bị quá tải về thực tế.

Để bảo đảm dữ liệu được nhận đúng, bên xuất dữ liệu và bên thu dữ liệu phải duy trì thời gian theo mối quan hệ như hình 1.4 đã chỉ ra. Giai đoạn 1 là giai đoạn xuất dữ liệu và ổn định dữ liệu trên kênh. Giai đoạn này là để bảo đảm tất cả các bit tin được tập kết đúng giá trị của mình trên kênh. Giai đoạn 2

là giai đoạn bện thu nhập dữ liệu từ kênh thông tin vào còn giai đoạn 3 là giai đoạn kết thúc việc xuất dữ liệu trên kênh, giải phóng kênh để chuẩn bị cho thao tác trao đổi dữ liệu tiếp theo.



Hình 1.4. **Đồ thị thời gian của tín hiệu điều khiển thu (IN) phát (OUT) trên kênh thông tin của hệ Vi xử lý**

1.3. BỘ NHỚ TRUNG TÂM CỦA HỆ VI XỬ LÝ

Bộ nhớ trung tâm là một bộ phận không thể thiếu đối với bất kỳ một hệ vi xử lý nào. Bộ nhớ trung tâm là tập hợp các thanh ghi thông tin với số lượng lớn. Mỗi thanh ghi có m bit (thường là 8 bit). Chức năng cơ bản của bộ nhớ là để lưu trữ và trao đổi thông tin.

Vấn đề quan trọng nhất trong phương pháp tổ chức bộ nhớ là phương thức quản lý các thanh ghi. Do số lượng các thanh ghi lớn nên phương thức hiệu quả nhất là quản lý theo phương thức ma trận điểm. Mỗi điểm là một thanh ghi. Một cấu trúc địa chỉ hoá như vậy cho phép thâm nhập vào bất kỳ thanh ghi nào của bộ nhớ mà không sợ nhầm lẫn. Chúng ta sẽ xét dạng cấu trúc này.

1.3.1. Quản lý bộ nhớ

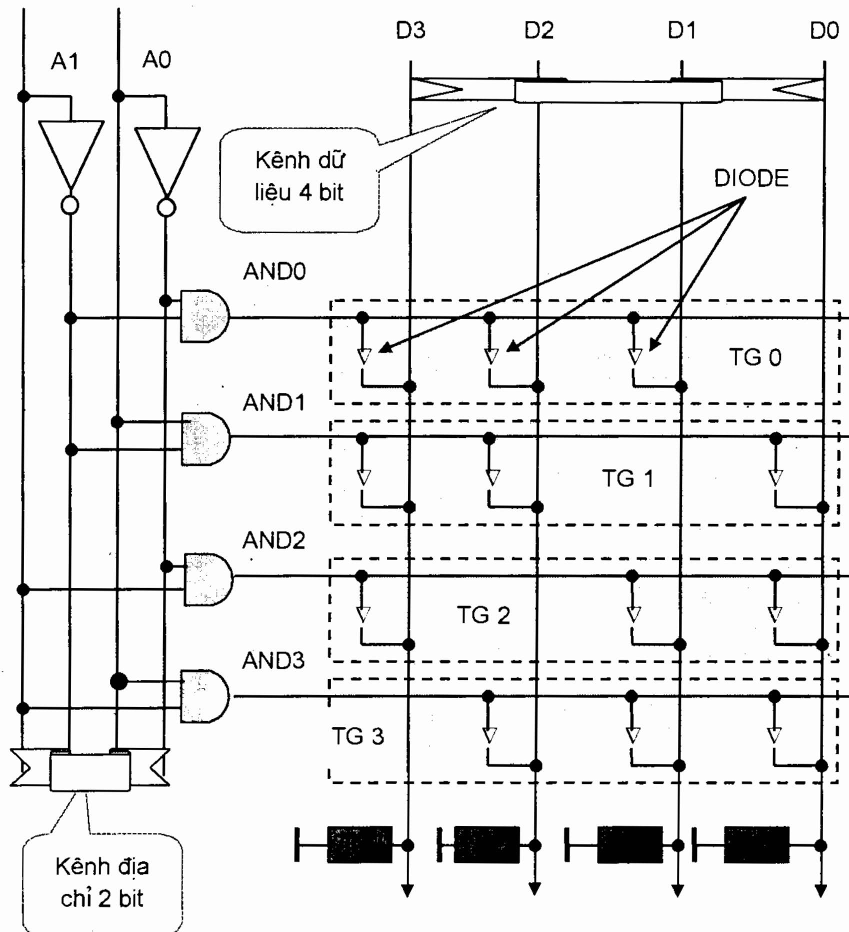
Sơ đồ trên hình 1.5 là một cấu trúc bộ nhớ điển hình cho phép ghi thông tin hoặc đọc thông tin từ bất kỳ thanh ghi nào của bộ nhớ. Bộ giải mã hàng biến m dây địa chỉ ở đầu vào thành 2^m hàng của ma trận quản lý.

Ngoài ra, ROM trong hệ vi xử lý còn dùng để lưu trữ các bảng, biểu, tham số hệ thống mà trong quá trình hoạt động không được thay đổi như bảng địa chỉ cổng giao tiếp, các bảng tra cứu số liệu, các bộ mã cần sử dụng trong hệ.

Phương thức quản lý bộ nhớ ROM cũng theo phương thức ma trận điểm. ROM có nhiều chủng loại khác nhau: ROM, PROM, EPROM ...

ROM là bộ nhớ cố định có cấu trúc đơn giản nhất. Nội dung của nó do nhà sản xuất chế tạo, người sử dụng không thể thay đổi nội dung này được nữa. Hình 1.6 biểu diễn sơ đồ cấu trúc của bộ nhớ ROM đơn giản bao gồm kênh địa chỉ 2 bit địa chỉ A1A0, kênh dữ liệu 4 bit D3D2D1D0. Về nguyên tắc, bộ giải mã địa chỉ sẽ cho phép quản lý được $2^2 = 4$ thanh ghi 4 bit, đó là các thanh ghi TG3 TG2 TG1 TG0.

Nội dung mỗi thanh ghi là số nhị phân có 4 chữ số. Mỗi chữ số là giá trị của của bit tương ứng của kênh dữ liệu. Giá trị này bằng 1 khi tại vị trí tương ứng của hàng và cột trên hình 1.6 có diode làm cầu dẫn. Tương tự như thế, giá trị này bằng 0 khi tại vị trí tương ứng của hàng và cột không có diode làm cầu dẫn. Bảng 1.1 liệt kê nội dung của các thanh ghi có trong ROM.



Hình 1.6. **Bộ nhớ ROM đơn giản 4 thanh ghi 4 bit**

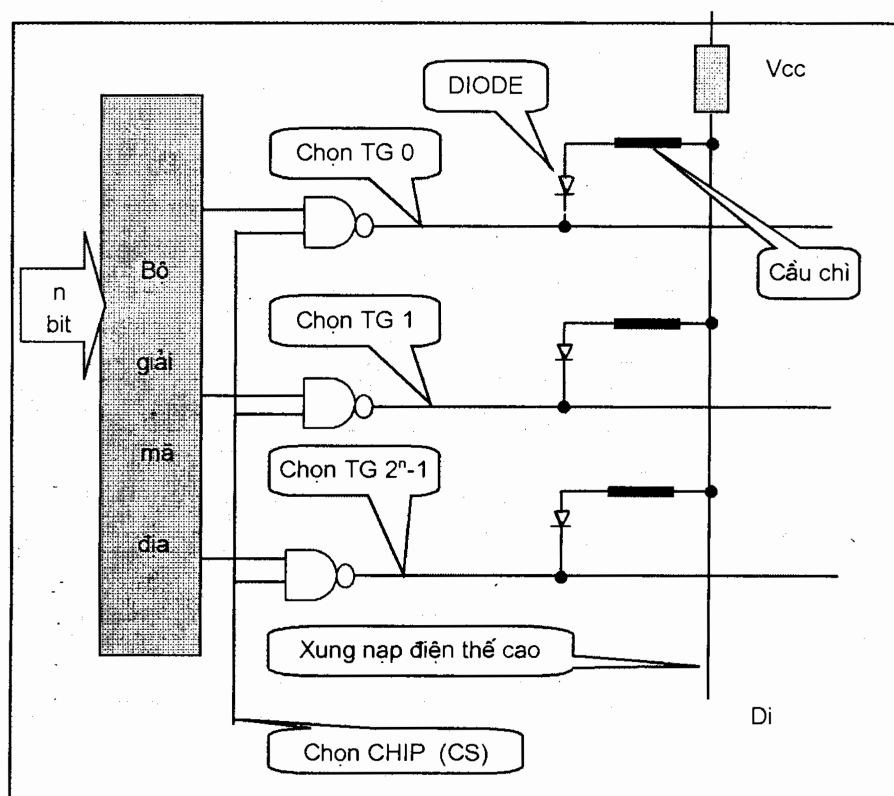
Bộ nhớ PROM (Programmable ROM- ROM chương trình hoá được) là bộ nhớ cố định có cấu trúc đơn giản. Nội dung của nó do nhà sản xuất hay người thiết kế hệ vi xử lý nạp vào nhưng chỉ được một lần. Sau khi nạp xong nội dung này không thể thay đổi được nữa.

Bảng 1.1

TG	Địa chỉ A1A0	Nội dung D3D2D1D0
R0	00	1 1 1 0
R1	01	1 1 0 1
R2	10	1 0 1 1
R3	11	0 1 1 1

Hình 1.7 biểu diễn sơ đồ cấu trúc của bộ nhớ PROM đơn giản bao gồm kênh địa chỉ n bit địa chỉ, kênh dữ liệu m bit. Về nguyên tắc, bộ giải mã địa chỉ sẽ cho phép quản lý được 2^n thanh ghi m bit.

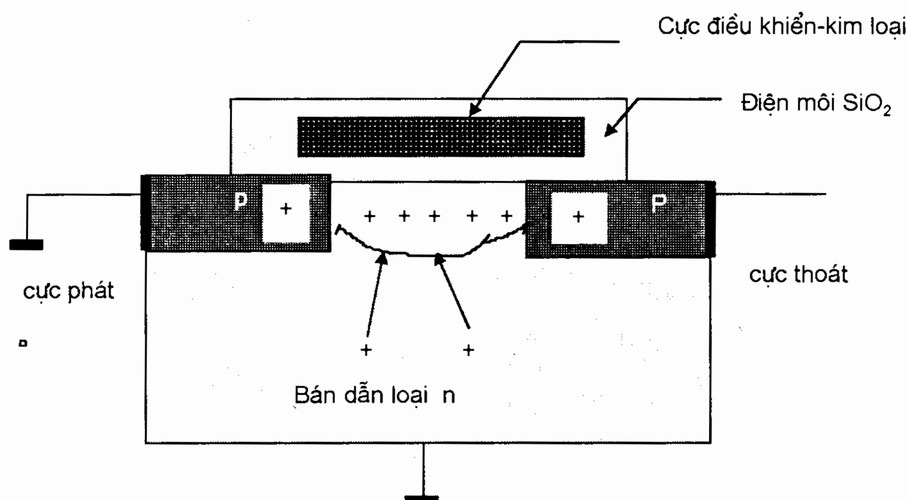
Nội dung mỗi thanh ghi là số nhị phân có m chữ số. Mỗi chữ số là giá trị của của D_i tương ứng của kênh dữ liệu. Giá trị này bằng 0 khi tại vị trí tương ứng của hàng và cột trên hình 1.7 có cầu dẫn. Tương tự như thế, giá trị này bằng 1 khi tại vị trí tương ứng của hàng và cột không có cầu dẫn.



Hình 1.7. **Cấu tạo và nguyên lý làm việc của bộ nhớ PROM**

Khi chế tạo, tất cả các vị trị giao nhau giữa hàng và cột đều có các cầu dẫn. Khi muốn tạo logic 1 ở bit nào đó của thanh ghi tương ứng, máy nạp chương trình sẽ cung cấp một xung dòng đủ lớn để đánh cháy cầu dẫn đi. Còn muốn tạo logic 0 ở bit nào đó của thanh ghi tương ứng, máy nạp chương trình sẽ không làm cháy cầu dẫn đó.

Bộ nhớ EPROM (Erasable PROM- ROM nạp/ xoá được nhiều lần) là bộ nhớ cố định có cấu trúc đặc biệt. Nội dung của nó do nhà sản xuất hay người thiết kế hệ vi xử lý nạp vào và có thể nạp/ xoá nhiều lần. Hình 1.8 biểu diễn sơ đồ cấu trúc của một bit thông tin dựa trên nguyên tắc làm việc của transistor trường MOS (Metall-Oxyde-Semiconductor) có cực điều khiển bị thả nổi.



Hình 1.8. **Một bit nhớ của EPROM bằng transistor MOS có cực điều khiển bị thả nổi.**

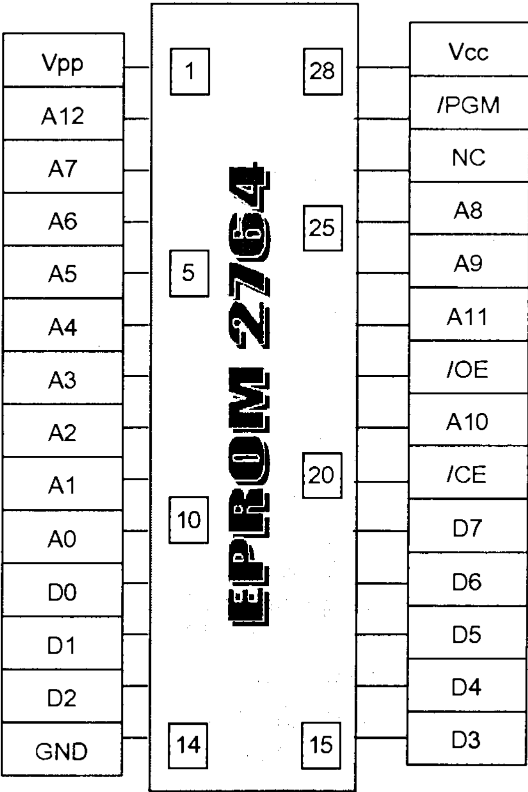
Khi muốn tạo logic 0 ở bit thông tin, cần tạo ra kênh cảm ứng giữa cực phát và cực thoát. Máy nạp chương trình sẽ tạo thiên áp ngược giữa hai cực gây ra dòng điện tử từ cực thoát vượt qua hàng rào thế năng của lớp điện môi SiO_2 vào lớp kim loại và bị giữ lại trong đó. Do trong lớp kim loại tích tụ một lượng điện tích âm đủ lớn nên nó tạo ra điện trường vuông góc với tấm đế bán dẫn, thu hút các động tử tải điện mang dấu dương vào lân cận hai cực, tạo thành kênh cảm ứng nối liền hai cực phát và thoát.

Khi muốn tạo logic 1 ở bit thông tin, máy nạp duy trì trạng thái cũ, tức là không thao tác gì cả. Điều này có nghĩa là một EPROM sạch là EPROM mà nội dung của nó chứa toàn giá trị logic 1 ở các bit.

Khi muốn xoá nội dung cũ của EPROM, người ta sử dụng tia cực tím có bước sóng tương ứng để chiếu xạ vào EPROM. Khi bị chiếu xạ, các điện tử trong lớp kim loại thu được năng lượng bổ xung và vượt qua hàng rào thế nằng của lớp điện môi rồi tiêu tán đi. Khi tích tụ này ở trong lớp kim loại giảm đi tới mức nào đó thì kênh cảm ứng cũng tiêu tán theo làm cho MOS trở về trạng đầu. Bằng cách đó ta xoá được nội dung của bộ nhớ EPROM.

1.3.3. Bộ nhớ IC thông dụng của ROM

Công nghệ vi điện tử đã tạo ra hàng loạt các bộ nhớ cố định ROM và EPROM dung lượng lớn dưới dạng các mạch IC. Chip nhớ EPROM thông dụng hiện nay là các IC nhớ có dung lượng từ 8 kb trở lên: 2764 (8 kb), 27128 (16 kb), 27256 (32 kb) ... Hình 1.9 là sơ đồ chân tín hiệu của bộ nhớ IC 2764 còn bảng 1.2 là chức năng làm việc của các tín hiệu đó.



Hình 1.9. **Chip IC EPROM 2764 (8kb)**

Bảng 1.2

MODE	/CE	/OE	/PGM	Vpp	Vcc	DATA BUS (11-13,15-19)
READ	L	L	H	Vcc	Vcc	output
chuẩn bị	H	X	X	Vcc	Vcc	High
nạp chương trình	L	X	L	Vpp	Vcc	input
kiểm tra ch.tr nạp	L	L	H	Vpp	Vcc	output
cấm nạp	H	X	X	Vpp	Vcc	hight

1.3.4. Bộ nhớ đọc/ghi RAM

Theo phương thức lưu trữ thông tin RAM được chia ra làm hai loại chính : RAM tĩnh và RAM động. RAM tĩnh có thể lưu trữ thông tin lâu tuỳ ý miễn là được cung cấp điện năng - tất cả các loại phần tử nhớ bằng trigơ đều thuộc loại này. RAM động, ngược lại, chỉ lưu được thông tin trong một thời gian nhất định. Muốn kéo dài thời gian này cần có phương thức làm tươi lại thông tin trong phần tử nhớ RAM. Phần tử nhớ của RAM động đơn giản nhất là một linh kiện điện dung - tụ điện. Sử dụng RAM động có phức tạp nhưng về cấu trúc nhớ lại đơn giản, tiêu tốn ít năng lượng, tăng mật độ bộ nhớ và đôi khi còn làm tăng cả tốc độ làm việc của bộ nhớ.

Cấu trúc mạch điện của các bộ nhớ RAM rất đa dạng cả về công nghệ chế tạo chúng (TTL, MOS, ECL, I^2L, SOS) và cả về các yêu cầu sử dụng chúng như các yêu cầu về ghép nối, tốc độ làm việc, mật độ linh kiện và dung lượng cần thiết ... Để minh hoạ ta sẽ xét các loại RAM tĩnh công nghệ TTL, MOS và RAM động công nghệ MOS.

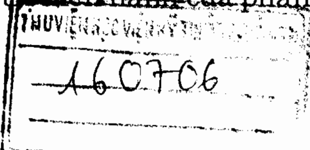
- ***Cấu trúc của bộ nhớ RAM tĩnh công nghệ MOS***

Trên hình 1.10 là cấu trúc một bộ nhớ RAM tĩnh (STATIC RAM) công nghệ MOS. Mỗi phần tử nhớ là hai tầng khuyếch đại ghép theo mạch hồi tiếp để tạo ra hai trạng thái ổn định (xem giáo trình kỹ thuật xung). Mỗi tầng là một transiztor MOS (là transiztor trường kênh n làm việc ở chế độ giầu động tử), cho phép ghép nối với các tín hiệu điều khiển (như chọn hàng, chọn cột của ma trận quản lý bộ nhớ) và các dây dữ liệu.

Dây dữ liệu được nối với cực nguồn của transizror MOS T4 của phần tử nhớ. Transiztor này bình thường ở trạng thái ngắt, cực máng của nó được nối với dây chọn hàng tương ứng của bộ giải mã địa chỉ. Tín hiệu điều khiển Write được nối với hai đầu vào của hai cổng AND, còn dây dữ liệu nguồn được nối với cổng AND thứ nhất qua cổng NOT và được nối trực tiếp với đầu vào thứ hai của cổng AND thứ hai.

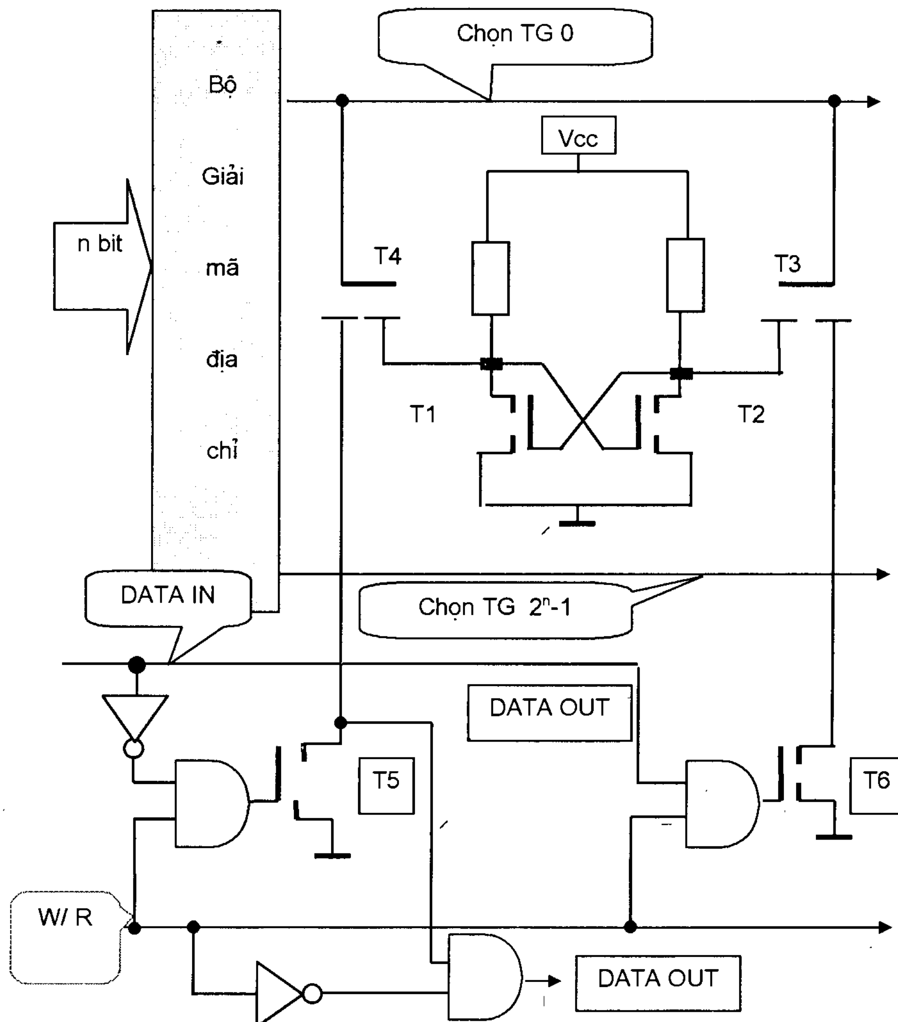
Khi tín hiệu điều khiển Wrire có mức lôgic 1 đồng thời thông tin trên dây dữ liệu cũng có mức lôgíc 1, cực điều khiển của T6 sẽ có điện thế cao làm T6 thông và do đó nửa phần bên phải của phần tử nhớ có mức thấp còn nửa phẩn bên trái có mức cao. Trạng thái này ứng với trạng thái lôgic 1. Trong trường hợp ngược lại, nếu thông tin trên dây dữ liệu có mức lôgic 0 còn tín hiệu điều khiển Write có mức lôgíc 1 sẽ làm T5 thông, nhờ đó phần tử nhớ chuyển sang trạng thái lôgic 0.

Nếu tín hiệu điều khiển Write có mức lôgic 0 thì cả hai tranzitor T5 và T6 đều ngắt và trạng thái của phần tử nhớ không thay đổi. Trên dây dữ liệu ra là trạng thái của phần tử nhớ, nó có thể được đọc ra khi đầu tín hiệu Write bằng lôgic 0 (DATA OUT bằng 1 hay 0 tuỳ thuộc vào giá trị hiện hành của phần tử nhớ).



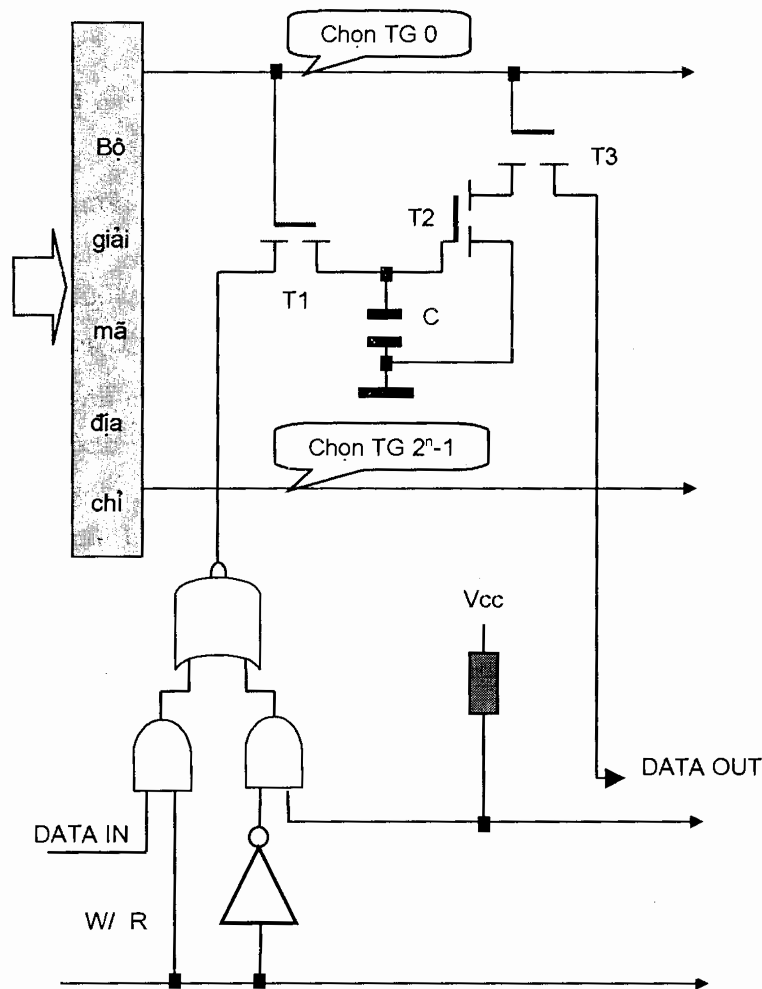
- ***Cấu trúc của bộ nhớ RAM động công nghệ MOS***

Bộ nhớ RAM động công nghệ MOS được thể hiện trên hình 1.11. Phần tử nhớ cơ bản của nó là một tụ điện và một transiztoz MOS T2. Tụ điện là một linh kiện độc lập, song trong thực tế người ta sử dụng luôn điện dung của cấu trúc cực điều khiển và tấm đế của tranzizitor này làm tụ điện.



Hình 1.10. **Cấu trúc và nguyên lý làm việc của RAM tĩnh (STATIC RAM) công nghệ MOS**

Thông tin được lưu trữ trong phần tử nhớ phụ thuộc vào điện dẫn của T2, tức là phụ thuộc vào lượng điện tích được tích luỹ trong tụ điện. Nếu điện tích được tích luỹ đủ lớn thì theo nguyên tắc làm việc của tranzistor trường có kênh cảm ứng loại n thì T2 sẽ thông.



Hình 1.11. **Bộ nhớ RAM động công nghệ MOS**

Điều cần lưu ý ở đây là trạng thái lôgíc 0 đạt được khi T2 thông lại là trạng thái không bền vì theo thời gian tụ điện sẽ phóng dần điện tích tích luỹ. Do điện dung của tụ điện rất nhỏ nên thời gian duy trì sẽ ngắn. Khi lượng điện tích bị tiêu hao quá một giá trị nào đó sẽ dẫn tới T2 sẽ ngắt. Trạng thái này tương ứng với trạng thái lôgíc 1 và nó là trạng thái bền. Muốn duy trì trạng thái 0 cần thường xuyên bổ xung điện tích cho tụ điện trong mạch. Quá trình này gọi là quá trình làm tươi thông tin.

Ngoài tụ điện và tranziztor T2 trong phần tử nhớ còn có hai tranziztor khác, đóng vai trò phần tử khoá để điều khiển việc xuất và nhập thông tin của phần tử nhớ. Tranziztor T1 là mạch khoá hai chiều, nó nối dây dữ liệu với tụ điện C. Nếu T1 thông, tụ điện trong mạch có thể được nạp hoặc phóng điện tích của mình qua dây dữ liệu. Tranzistor T3 cũng là một mạch khoá nối dây dữ liệu

ra với cực thoát của T2. Nếu T3 thông, trạng thái lưu trữ trong phần tử nhớ sẽ được truyền qua T3 để dẫn ra ngoài bằng dây dữ liệu ra. Chế độ làm việc của T2 và T3 được điều khiển bằng tín hiệu chọn hàng của bộ giải mã địa chỉ hàng.

Tín hiệu điều khiển Write sẽ điều khiển chế độ làm việc của các dây dữ liệu vào và dữ liệu ra. Cần chú ý thêm là ở đây giá trị dữ liệu vào luôn luôn bị đảo pha vì mức điện áp cao trên tụ điện lại ứng với mức lôgíc 0, điều mà bị coi là ngược với những khái niệm quen thuộc được chấp nhận trước đây.

Điện dẫn của T2 được thể hiện ở mức điện áp trên dây dữ liệu ra nhờ điện trở tải R ở mạch ngoài. Nếu T2 thông, nghĩa là phần tử nhớ ở trạng thái lôgíc 0, trên dây dữ liệu ra sẽ có mức điện thế thấp. Nếu T2 ngắt, nghĩa là phần tử nhớ ở trạng thái lôgíc 1 vì điện trở R mắc vào nguồn điện áp nên trên dây dữ liệu ra sẽ có mức điện thế cao.

1.3.5. Bộ nhớ IC thông dụng của RAM

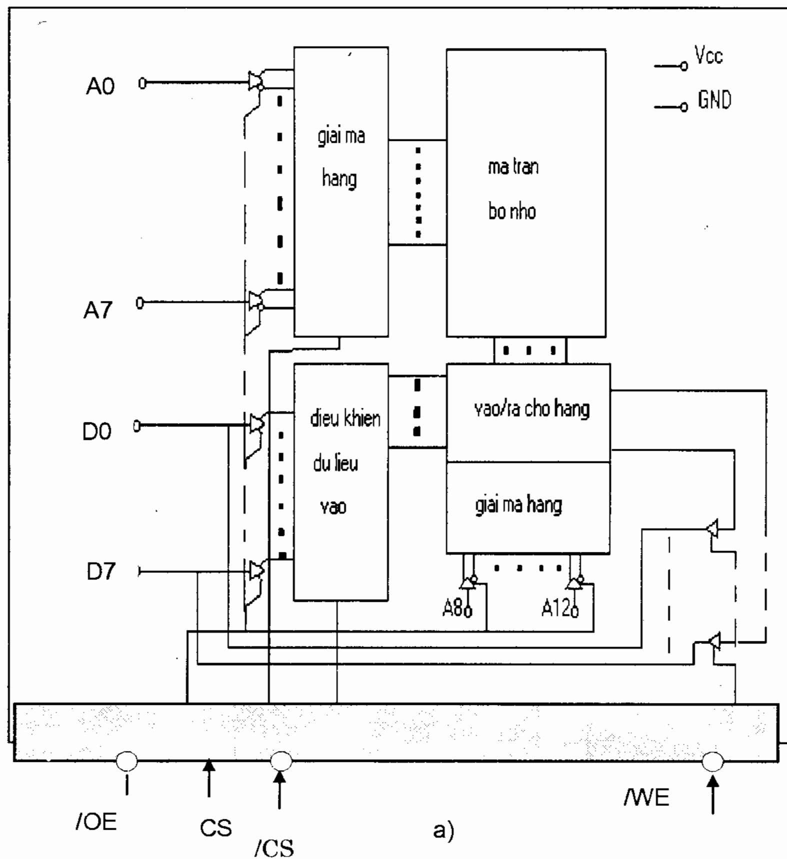
Công nghệ vi điện tử đã tạo ra hàng loạt các bộ nhớ cố định RAM tĩnh và RAM động dung lượng lớn dưới dạng các mạch IC. Chip nhớ RAM tĩnh thông dụng hiện nay là các IC nhớ có dung lượng từ 8 kb 6264. Hình 1.12a là sơ đồ chức năng làm việc của các khối chính còn hình 1.12b là sơ đồ chân tín hiệu của bộ nhớ và bảng 1.3 là các chế độ (MODE) làm việc của IC 6264.

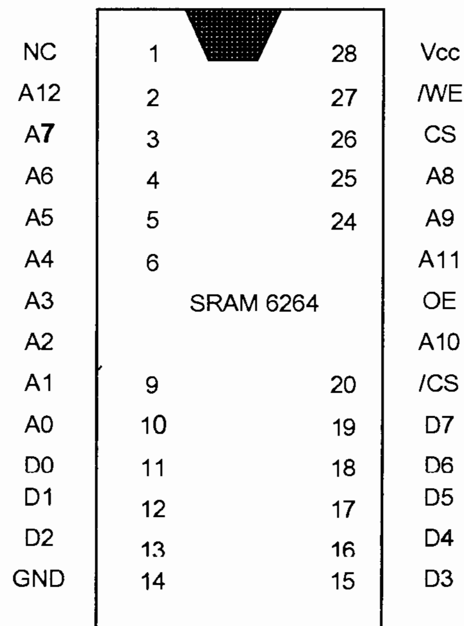
1.4. TỔ CHỨC BỘ NHỚ TRUNG TÂM CỦA HỆ VI XỬ LÝ

Bộ nhớ trung tâm của hệ vi xử lý thường là không đủ dung lượng cần thiết nếu chỉ sử dụng một vài IC nhớ.Trong những trường hợp như thế chúng phải được ghép nối với nhau để tạo dung lượng cần thiết. Tuỳ thuộc vào yêu cầu về dung lượng, về độ dài từ mã và loại IC nhớ mà có nhiều kiểu ghép nối khác nhau. Sau đây chúng ta xét các kiểu ghép nối đó.

1.4.1. Tổ chức bộ nhớ trung tâm kiểu ghép song song các IC nhớ

Nếu dung lượng ngăn nhớ của IC đủ, còn độ dài của ngăn nhớ tính theo bit không đủ thì cần tổ chức ghép song song nhiều IC lại với nhau. Kênh được dùng song song ở đây là kênh địa chỉ và kênh điều khiển. Thí dụ, nếu muốn có bộ nhớ RAM 8 K ngăn nhớ, mỗi ngăn nhớ có độ dài 64 bit (kênh dữ liệu 64 bit) thì ta ghép 8 IC RAM 6264 theo kiểu ghép song song như hình 1.13 đã chỉ ra.

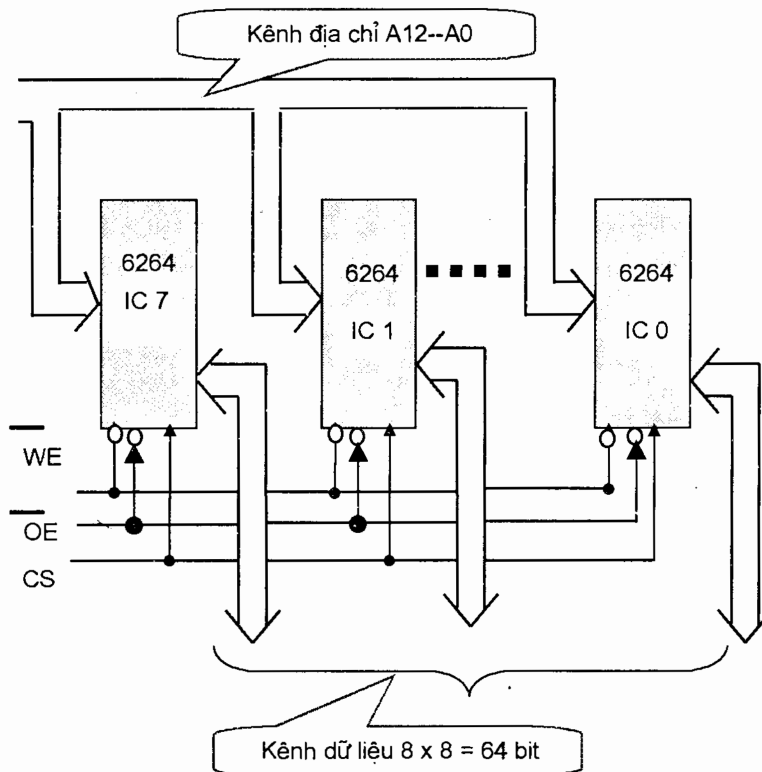




Hình 1.12. **RAM tĩnh 6264 (8KB):** a) Sơ đồ chức năng. b) Sơ đồ chân tín hiệu.

Bảng 1.3

$\overline{WE}$	$\overline{CS}$	CS	$\overline{OE}$	Không chọn chip
X	H	X	X	MODE
H	L	H	H	Không xuất dữ liệu
H	L	H	L	Đọc
L	L	H	L	Ghi

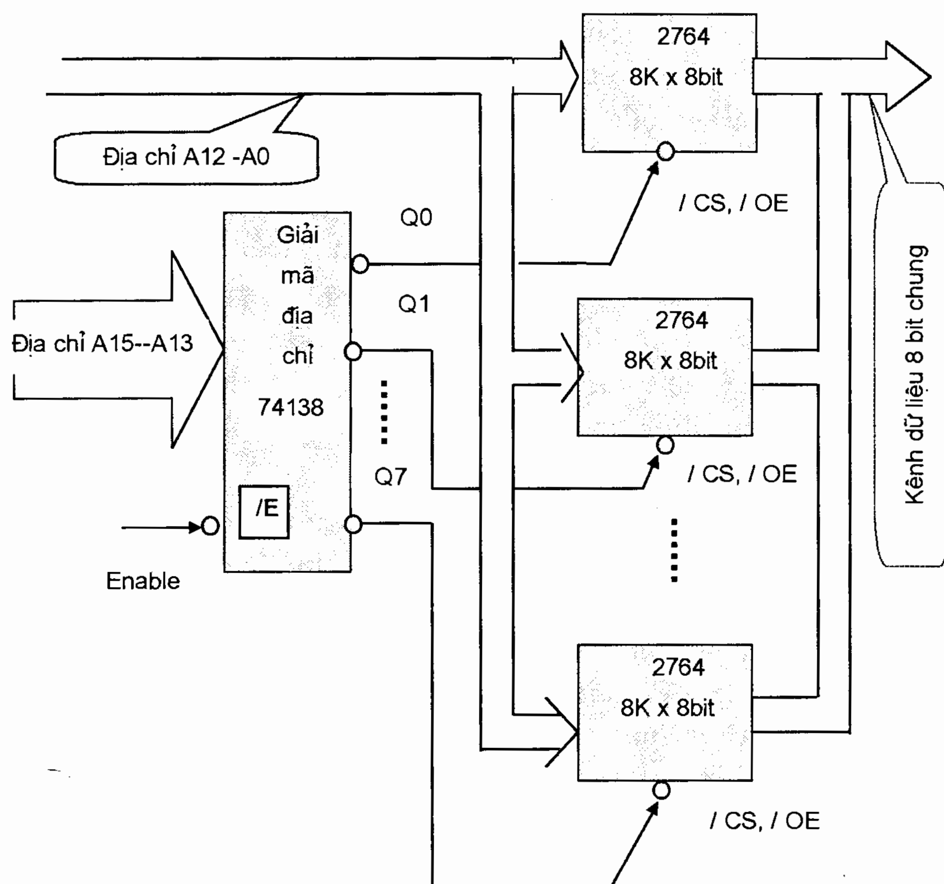


Hình 1.13. **Tổ chức bộ nhớ RAM song song dung lượng 8Kx64bit từ 8 chíp IC 6264.**

1.4.2. Tổ chức bộ nhớ trung tâm kiểu ghép nối tiếp các IC nhớ

Nếu dung lượng ngăn nhớ của IC không đủ, còn độ dài của ngăn nhớ tính theo bit đủ thì cần tổ chức ghép nối tiếp nhiều IC lại với nhau. Khi đó đòi hỏi phải có những mạch mở rộng địa chỉ. Đó là một mạch giải mã địa chỉ cho phép

chọn vi mạch này hay vi mạch khác. Để tạo được những bit địa chỉ phụ người ta thường sử dụng các bộ demutiplexer n đầu vào và 2^n đầu ra. Những dây địa chỉ phụ này được nối với đầu vào của bộ giải mã, đầu ra của chúng cung cấp các tín hiệu chọn các mạch nhớ. Bản thân các bộ giải mã này cũng có những dấu hiệu chọn mạch để điều khiển chế độ làm việc của chúng.



***Hình 1.14.* Tổ chức bộ nhớ nối tiếp dung lượng 8Kx8bit từ 8 chip 1K x8bit**

Nếu bộ giải mã ở chế độ cấm – thì tất cả các đầu ra của nó có mức logic cao- mức thụ động. Nếu bộ giải mã ở chế độ làm việc - một trong các đầu ra sẽ có mức lôgic thấp - mức tích cực. Các bộ giải mã thông dụng như 74138 hoặc 8205 của Intel đều là những mạch giải mã có 3 đầu vào và 8 đầu ra (Hình 1.15). Hình 1.14 là một phương án sử dụng bộ giải mã 74138 để điều khiển bộ nhớ EPROM có dung lựng 64Kb từ 8 mạch IC có dung lượng 8Kb 2764.

Khi các IC nhớ được tổ hợp lại thành bộ nhớ có số lượng từ mở rộng thì những đầu ra dữ liệu tương ứng của từng IC nhớ phải được dồn lại để tạo thành một kênh chung có thể hoà vào kênh dữ liệu của hệ thống. Nếu các bộ đệm ra của mỗi IC nhớ là mạch hở Colectơ hay mạch ba trạng thái thì có thể dồn các bít dữ liệu tương ứng lại với nhau mà không cần mạch lôgíc phụ. Những đầu hở colectơ có thể nối chung và chúng vẫn thực hiện đúng chức năng lôgic của mình.

Tuy nhiên cần lưu ý là chúng không phải là đầu ra lôgíc chuẩn. Khi các tín hiệu giải mã có tác dụng, các cổng ba trạng thái của một trong các IC nhớ sẽ được mở tương ứng đúng với tổ hợp của các tín hiệu điều khiển và địa chỉ đầu vào của nó.

Một số các IC nhớ kiểu RAM chỉ có một kênh dữ liệu chung cho cả kênh ra và kênh vào, còn một số IC nhớ kiểu RAM khác lại có kênh vào và kênh ra tách biệt. Cả hai trường hợp trên, bộ RAM có ít nhất một đầu điều khiển để khống chế thời điểm mở các cổng đệm 3 trạng thái ở kênh dữ liệu ra. Như vậy, đối với các bộ nhớ có chung một kênh dữ liệu vào/ra, ngoài tín hiệu điều khiển chọn IC (***chip select***) cần có thêm các tín hiệu kiểm soát khác (cấm mở đầu ra). Khi một IC nhớ nào đó được chọn để ghi thông tin vào, đầu ra của chúng phải ở trạng thái trở kháng cao nhờ có tín hiệu điều khiển này (cấm mở đầu ra).

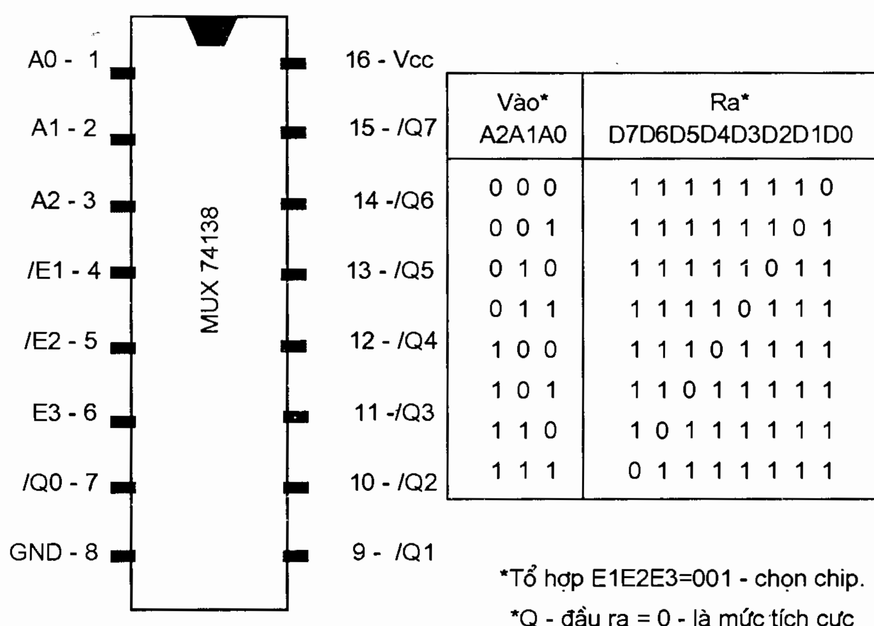
Đối với các bộ nhớ có kênh dữ liệu vào/ra tách biệt, đầu ra của kênh được mở khi có tín hiệu ***chip select***. Những kênh này đôi khi ghép qua bộ đệm bên ngoài để hoà chung vào kênh dữ liệu hai chiều của hệ thống. Tín hiệu điều khiển để mở các bộ đệm đó tương tự như tín hiệu cho phép mở của Tecxas Instruments hoặc INS 8202 của National.

Một bộ nhớ có dung lượng bất kỳ có thể được thiết kế để đáp ứng một yêu cầu kỹ thuật nào đó. Các chủng loại bộ nhớ khác như EPROM, PROM hoặc RAM... có thể kết hợp xen kẽ với nhau chiếm những vùng nhớ của hệ vi xử lý. Bộ nhớ cũng không nhất thiết phải chiếm hết toàn bộ không gian địa chỉ và cũng không nhất thiết phải xắp xếp tuần tự. Trong các hệ vi xử lý có thể tồn tại một vài vùng nhớ để trống. Cần chú ý rằng khi các IC nhớ đã được tổ chức thành bộ nhớ của hệ vi xử lý thì thời gian thâm nhập, thời gian để ghi nhận thông tin cho một IC nhớ có thể lớn hơn so với thời gian tương ứng cho một IC nhớ. Đó là do tồn tại các thời gian trễ ở những khâu trung gian và do các điện dung, điện cảm ký sinh trong quá trình lắp ráp các IC và các linh kiện điện tử khác.

1.4.3. Đồ thị thời gian của bộ nhớ

Sự lan truyền tín hiệu trong các phần tử, dù là các phần tử vi điện tử, là hữu hạn, tức là phải tiêu tốn một lượng thời gian nhất định để thông tin có thể truyền từ một bộ phận này sang bộ phận khác. Chính vì hạn chế này nên các tín hiệu trên các kênh thông tin của bộ nhớ phải tuân theo một trình tự thời gian nhất định.

Để mô tả trình tự thời gian của các tín hiệu, chúng ta coi bộ nhớ là lý tưởng tức là không có thời gian dữ chậm trong. Gọi Ta là thời gian giữ chậm của tín hiệu trên các dây địa chỉ khi lan truyền qua bộ đệm vào. Các bộ giải mã hàng/cột và qua Tcs mạch khoá đặc trưng của phần tử nhớ. Tcs là thời gian giữ chậm của tín hiệu điều khiển chip select khi phải lan truyền qua các cổng chọn mạch. To là thời gian giữ chậm của dữ liệu ra.



Vào* A2A1A0	Ra* D7D6D5D4D3D2D1D0
0 0 0	1 1 1 1 1 1 1 0
0 0 1	1 1 1 1 1 1 0 1
0 1 0	1 1 1 1 1 0 1 1
0 1 1	1 1 1 1 0 1 1 1
1 0 0	1 1 1 0 1 1 1 1
1 0 1	1 1 0 1 1 1 1 1
1 1 0	1 0 1 1 1 1 1 1
1 1 1	0 1 1 1 1 1 1 1

*Tổ hợp E1E2E3=001 - chọn chip.

*Q - đầu ra = 0 - là mức tích cực

Hình 1.15. **Bộ giải mã 3 đầu vào 8 đầu ra 74138**

Căn cứ vào sơ đồ ở hình 1.16 từ thời điểm xuất hiện một địa chỉ xác định ở kênh địa chỉ tới lúc dữ liệu ra đã ổn định trên kênh dữ liệu sẽ mất một khoảng thời gian là:

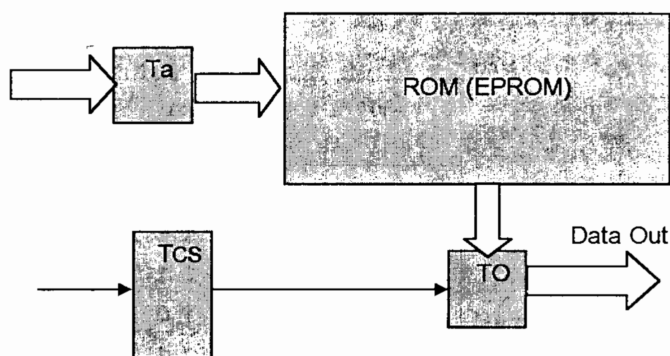
T4 = Ta + To

Nếu một giá trị địa chỉ đã ổn định trên kênh địa chỉ và tín hiệu chip select thay đổi để chọn bộ nhớ ROM thì thời điểm trễ kể từ thời điểm xuất hiện tín hiệu chip selec đến thời điểm dữ liệu ra ổn định trên kênh dữ liệu được tính bằng giá trị:

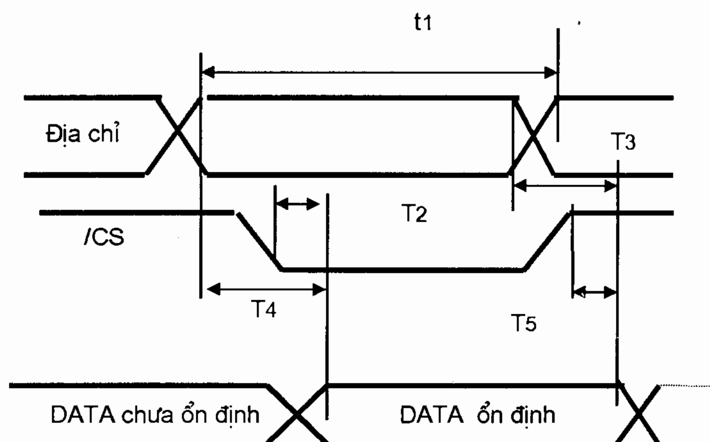
t2 = Tcs + To

Thời gian giữ chậm t2 thường nhỏ hơn Ta vì khối điều khiển ***chip select*** được nối trực tiếp với bộ đệm ra. Hình 1.16 b là đồ thị thời gian của các tín hiệu vừa xét trên.

Nói chung thời gian cần thiết để từ khi có tín hiệu địa chỉ tới lúc có dữ liệu ra ổn định được gọi là thời gian thâm nhập. Ba tham số khác là t3, t5, t1 cho biết dữ liệu còn ổn định được bao lâu sau khi địa chỉ đã thay đổi, t5 là thời gian ổn định của dữ liệu khi tín hiệu CS đã hết tác dụng (vào chế độ thả nổi). t1 là thời gian của chu kỳ đọc, nó xác định tốc độ cực đại của việc xuất thông tin từ ROM ra kênh dữ liệu.



a)



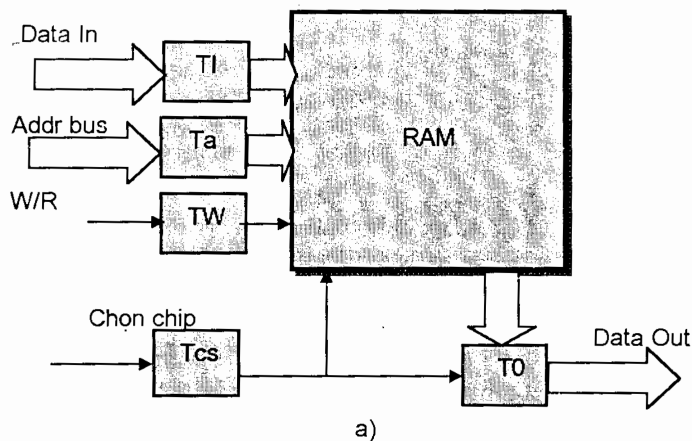
b)

Hình 1.16. a) Độ trễ thời gian và b) đồ thị gian trễ của ROM (EPROM)

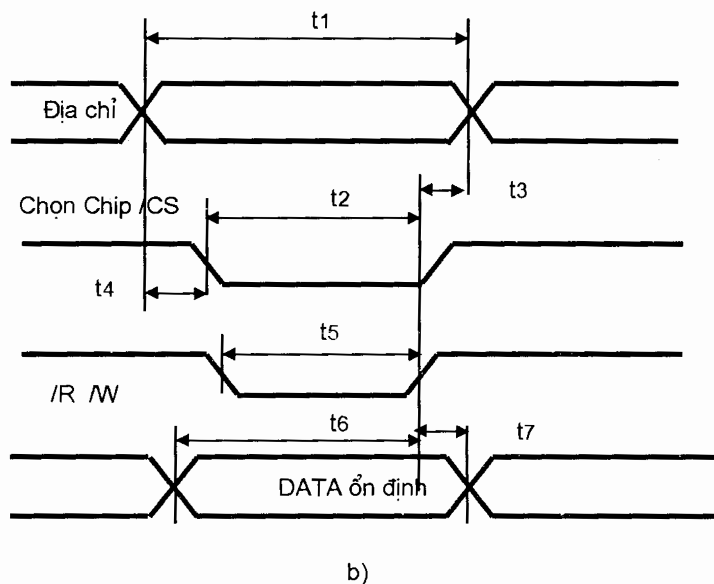
Đối với các bộ nhớ RAM. ngoài kênh dữ liệu và kênh địa chỉ giống như bộ nhớ ROM, RAM còn đòi hỏi một tập hợp các tín hiệu điều khiển việc xuất nhập thông tin vào RAM, ví dụ nếu tín hiệu điều khiển R/W=1: xuất thông tin thì R/W = 0 sẽ là nhập thông tin vào RAM. Do cấu trúc mạch phức tạp hơn ROM nên đối với RAM, các tín hiệu trên các kênh càng phải tuân thủ theo trình tự thời gian chặt chẽ hơn.

Mô hình về các khâu giữ chậm của bộ nhớ RAM được thể hiện ở hình 1.17. So với mô hình 1.16, mô hình này có thêm hai khâu dữ chậm là T1 - thời gian giữ chậm dữ liệu vào và TW - thời gian giữ chậm của thao tác ghi thông tin.

Tín hiệu điều khiển thao tác ghi thông tin vào RAM thông thường là một xung cực tính âm và phải tồn tại trong khoảng thời gian tối thiểu là t5 để bảo đảm chắc chắn cho việc ghi, dù là đối với bộ nhớ có tốc độ làm việc thấp nhất.



a)



b)

Hình 1.17. a) Độ trễ thời gian và b) đồ thị thời gian trễ của RAM

Như vậy thao tác ghi thông tin bắt đầu từ thời điểm xẩy ra quá trình quá độ từ mức logíc 1 xuống mức lôgíc 0 của tín hiệu R/W. Tín hiệu này phải mất một thời gian là TW để thâm nhập vào bộ nhớ, còn dữ liệu vào cũng phải mất một thời gian T1 để lan truyền tới đầu vào phần tử nhớ. Như vậy dữ liệu cần phải được ổn định ít nhất là trong khoảng thời gian t6 = (T1-TW) trước khi xẩy ra quá độ từ mức lôgíc 0 lên mức lôgíc 1 của tín hiệu điều khiển R/W. Dữ liệu cũng phải giữ được ổn định trong một thời gian sau khi xung R/W kết thúc. Giá trị t3 là thời gian hồi phục thao tác ghi, nó cho biết thời gian cần giữ cho địa chỉ ổn định sau khi xung R/W kết thúc.

Cũng như đối với các đầu vào kênh địa chỉ, tín hiệu ***chip select*** cần ổn định trước khi xuất hiện tín hiệu R/W. Thời gian này được gọi là t1, tham số t1 là chu kỳ ghi thông tin. Đối với phần lớn các bộ nhớ RAM tĩnh chu kỳ xuất và chu kỳ nhập thông tin là như nhau.

CHƯƠNG 2
BỘ VI XỬ LÝ 16 BIT 80286 INTEL

Trong các thành phần của hệ vi xử lý đã xét ở chương 1 thì bộ vi xử lý là thành phần quan trọng nhất. Các bộ vi xử lý được nhiều hãng sản xuất chế tạo và phát triển, nhưng người để lại dấu ấn quan trọng nhất là hãng INTEL. Các hệ vi xử lý của hãng này, từ hệ 8 bit đến 16/32 bit luôn luôn chiếm thị phần quan trọng của nhiều lĩnh vực kỹ thuật. Chương này sẽ nghiên cứu nguyên lý làm việc của các bộ vi xử lý, bắt đầu từ bộ vi xử lý 16 bít 80286 Intel nhằm làm cơ sở cho phần thiết kế ra các hệ vi xử lý chuyên dụng, phục vụ cho các yêu cầu cụ thể của kỹ thuật Điện tử - Viễn thông hay kỹ thuật Tự động điều khiển. Khi đã làm chủ được bộ vi xử lý 16 bít 80286 sẽ dễ dàng tiếp cận được các bộ vi xử lý cao cấp hơn như 80586 rồi tới PENTIUM và hơn nữa trong tương lai. Bộ vi xử lý 16 bit 80286 Intel là bộ vi xử lý cao cấp đầu tiên được trang bị cơ chế quản lý địa chỉ ảo là cơ chế cho phép chạy trên hệ điều hành đa nhiệm. Đây chính là bước tiến bộ nhảy vọt về chất trong cấu trúc của các bộ vi xử lý. Khi làm việc, hoạt động được trong chế độ đa nhiệm, các bộ vi xử lý huy động được tối đa khả năng của mình trong các chức năng xử lý song song, sử dụng tài nguyên chung, mở rộng không gian nhớ ra ngoài không gian nhớ thực của chính bộ vi xử lý đó.

2.1. TỔ CHỨC PHẦN CỨNG CỦA BỘ VI XỬ LÝ 80286

2.1.1. Cấu trúc chung của bộ vi xử lý 80286

Bộ vi xử lý nói chung có tên tiếng Anh là MICROPROCESSOR (còn gọi tắt là MP) hoặc CENTRAL PROCESSING UNIT (còn gọi tắt là CPU). Bộ vi xử lý 80286 Intel là bộ vi xử lý 16 bit nằm trong họ vi xử lý IAPX 86. Những đặc trưng cơ bản của 80286 là:

- Tần số xung nhịp là 16MHz.
- Khả năng địa chỉ hoá được 16 MB bộ nhớ vật lý và 1GB nhớ ảo.
- Có hai chế độ hoạt động là chế độ thực và chế độ ảo (chế độ bảo vệ).
- Có khả năng làm việc với các bộ đồng xử lý.

Trong chế độ địa chỉ thực, 80286 có mã lệnh tương thích với tập lệnh của các bộ vi xử lý 8086 và 8088 nhưng tốc độ làm việc nhanh hơn do tần số xung nhịp đồng bộ của 80286 cao hơn (có thể đạt tới 16MHz). Khả năng địa chỉ hoá bộ nhớ vật lý lớn nhất là 16MB (2^{24}). Tuy nhiên, trong chế độ địa chỉ thực, 80286

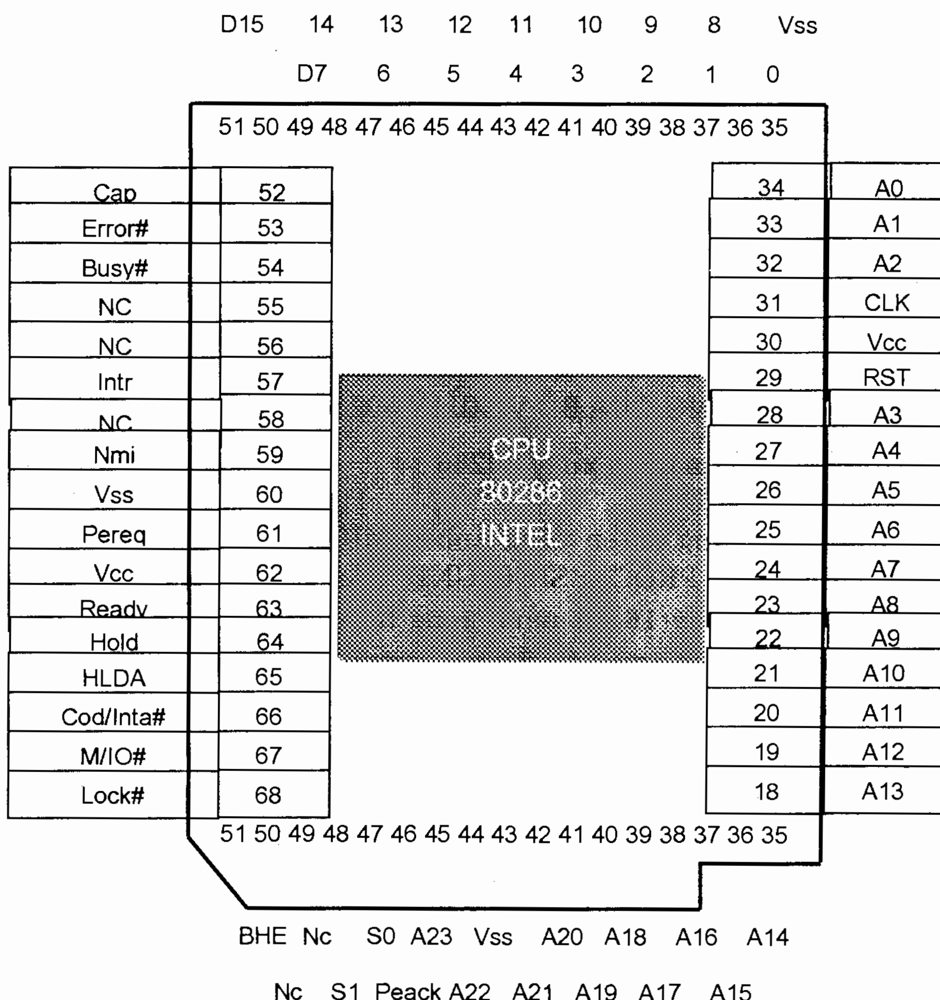
chỉ sử dụng các đường địa chỉ A_0 tới A_{19} để đánh địa chỉ nên có thể quản lý được 1MB (2^{20}) ngăn nhớ.

Trong chế độ địa chỉ ảo, bộ vi xử lý 80286 có mã gốc tương thích với tập lệnh của 8086 và 8088. Khả năng địa chỉ hoá ở chế độ này là 16MB bộ nhớ vật lý và 1GB nhớ ảo (2^{30}). Chế độ này thực hiện được cơ chế bảo vệ bộ nhớ và sử dụng đơn vị quản lý bộ nhớ ở bên trong bản thân bộ vi xử lý 80286.

Các tín hiệu của 80286 được phân theo nhóm và được tổ chức như ở hình 2.1. ***CLK*** là tín hiệu xung đồng hồ cho bộ vi xử lý. Tần số vào CLK sẽ được chia đôi ở bên trong 80286. Như vậy, một chu kỳ PCLK (peripheral CLK) được tạo thành từ hai chu kỳ CLK. Mỗi CLK được gọi là một pha của 80286 (φ1 và φ2).

Cap là chân gắn tụ lọc nhiễu. Tụ điện này có trị số 0,047 μF ± 20%, 12V được nối giữa chân CAP (chân 52) với đất để khử nhiễu xung có độ rộng hẹp của nguồn điện nuôi.

$\boldsymbol{D_0}$ tới $\boldsymbol{D_{15}}$ là 16 bít của kênh dữ liệu hai chiều, ba trạng thái và có mức tích cực là logíc 1.



Hình 2.1. **Tổ chức chân tín hiệu của bộ vi xử lý 80286**

A_0 tới A_{23} là kênh địa chỉ. Kênh này gồm 24 dây địa chỉ dùng để địa chỉ hoá cho các bộ nhớ vật lý và cho các cổng vào/ra. Dung lượng bộ nhớ có thể địa chỉ hoá trực tiếp được là 2^{24}=16M byte. Khi làm việc với các cổng vào/ra thì các chân địa chỉ A16 tới A23 bị ghim ở mức logíc 0 còn A0 tới A15 được sử dụng, do đó 80286 có thể quản lý trực tiếp được 64 KB cổng vào/ra.

/BHE là tín hiệu cho phép chọn phần cao của kênh dữ liệu. M/IO# (*dấu thăng # và / là mức tích cực âm*) được sử dụng để phân biệt quá trình quy chiếu bộ nhớ với quá trình quy chiếu cổng vào/ra. Tín hiệu /BHE kết hợp với A_0 cho biết dữ liệu được truyền theo byte hay word.

Mã của các phương thức truyền được cho bởi bảng 2.1.

$\overline{\text{BHE}}$	A0	Chức năng
0	0	Dữ liệu—WORD D15-D0
0	1	Dữ liệu—BYTE cao D15-D8
1	0	Dữ liệu—BYTE thấp D7-D0
1	1	Không sử dụng

M/IO# là tín hiệu dùng để phân biệt bộ nhớ và cổng vào/ra thông tin khi mà bộ vi xử lý cần quy chiếu chúng. Nếu quy chiếu bộ nhớ (tương ứng với chu kỳ đóng Shutdown) thì tín hiệu M/IO# = 1, ngược lại khi bộ vi xử lý 80286 quy chiếu cổng vào/ra thông tin thì tín hiệu này M/IO# = 0 (kể cả chu kỳ trả lời ngắt).

Bảng 2.2

COD/INTA#	M/IO#	S1	S0	Chu kỳ máy của 80286
0	0	0	0	Trả lời ngắt
0	0	0	1	Chưa sử dụng
0	0	1	0	Chưa sử dụng
0	0	1	1	Không là chu kỳ máy
0	1	0	0	A1=1 dừng A1=0 đóng
0	1	0	1	Chu kỳ máy MR
0	1	1	0	Chu kỳ máy MW
0	1	1	1	Không là chu kỳ máy
1	0	0	0	Chưa sử dụng
1	0	0	1	Chu kỳ máy IOR
1	0	1	0	Chu kỳ máy IOW
1	0	1	1	Chưa sử dụng
1	1	0	0	Chưa sử dụng
1	1	0	1	Chu kỳ máy OF
1	1	1	0	Chưa sử dụng
1	1	1	1	Không là chu kỳ máy

Phải lưu ý là từ bộ vi xử lý 80286 trở đi (đến Pentium) thì chức năng đa nhiệm (multitasking) mới thực sự được cài đặt nhờ chế độ quản lý không gian nhớ ảo của nó. Bộ nhớ ảo (VIRTUAL MEMORY) cho phép hệ vi xử lý làm việc với bộ nhớ mà dung lượng của nó lớn hơn rất nhiều so với bộ nhớ thực hiện có. Trong phần tiếp theo sẽ trình bày kỹ chế độ quản lý bộ nhớ kiểu này.

COD/INTA# cùng với các tín hiệu ***M/IO#***, ***/S_1***, ***/S_0*** cho phép phân biệt chu kỳ nhận lệnh và chu kỳ đọc dữ liệu từ bộ nhớ. Nó cũng được dùng để xác định chu kỳ trả lời ngắt, chu kỳ đọc cổng vào/ra và chu kỳ ghi cổng vào/ra. /S1 và /S0 sẽ xác định thời điểm ban đầu của chu kỳ máy.

Bảng 2.3

Các tín hiệu ra	Trạng thái
/S0,/ S1, /PEACK, A23--A0	1
/BHE, /LOCK	1
M/IO#, COD/INTA#, HLDA	0
D15--D0	Trở kháng cao

/READY là tín hiệu sẵn sàng làm việc của thiết bị ngoại vi. ***INTR*** là tín hiệu yêu cầu ngắt của thiết bị ngoại vi. Yêu cầu ngắt này sẽ bị che nếu bit IF ở thanh ghi cờ có giá trị logíc 0.

Bộ vi xử lý 80286 trả lời ngắt bằng hai chu kỳ /INTA: một chu kỳ thông báo cho mạch điều khiển ngắt (ví dụ 8259) biết là yêu cầu ngắt đã được chấp nhận và chu kỳ thứ hai sẽ đọc vectơ ngắt từ mạch điều khiển ngắt. Tín hiệu INTR phải ở mức tích cực ít nhất là hai chu kỳ trước lúc kết thúc lệnh đang thực hiện và phải duy trì cho đến cuối chu kỳ /INTA thứ nhất.

NMI là tín hiệu ngắt không che được. Nó phản ứng với sườn lên của xung kích. Lưu ý rằng muốn ngắt bằng tín hiệu này thì xung kích phải kéo dài mức logic 0 đúng 4 chu kỳ nhịp rồi mới chuyển sang mức 1 và duy trì mức này 4 chu kỳ nhịp nữa.

HOLD và HLDA là tín hiệu phục vụ cho chế độ thâm nhập trực tiếp DMA (Direct Memory Access).

LOCK là tín hiệu cấm các bộ vi xử lý khác (trong chế độ làm việc song song) dành quyền làm chủ kênh hệ thống. Lệnh LOCK khởi động tín hiệu trên và nó sẽ trở thành mức tích cực một cách tự động khi thực hiện lệnh XCHG hoặc ở chu kỳ /INTA hoặc trong thời gian thâm nhập bảng các bộ mô tả.

PEREQ và /PEACK phục vụ chế độ đối thoại giữa bộ vi xử lý 80286 với các bộ đồng xử lý ví dụ như bộ đồng xử lý 80287. Tín hiệu PEREQ ở mức logíc 1 là tín hiệu yêu cầu 80286 gửi một toán hạng cho bộ đồng xử lý. Tín hiệu /PEACK thông báo cho bộ đồng xử lý biết toán hạng mà nó yêu cầu đang được chuyển tới.

/BUSY và /ERROR là tín hiệu điều khiển cho bộ đồng xử lý. /BUSY thông báo cho 80286 biết rằng bộ đồng xử lý đang bận. Lúc này bộ vi xử lý 80286 thực hiện các lệnh ESC và WAIT để đợi bộ đồng xử lý. Tín hiệu /ERROR sẽ báo cho 80286 biết rằng bộ đồng xử lý phát hiện điều kiện ngoại lệ không che được.

RESET là tín hiệu khởi động có mức tích cực là mức logíc 1. Tín hiệu này sẽ lập trạng thái ban đầu cho tất cả các thành phần cần thiết của hệ vi tính. Khi có RESET thì các tín hiệu ra có mức logíc như sau:

Các tín hiệu ra	Trạng thái
/S1, /S0, /PEACK, A0-A23	1
/BHE, /LOCK	1
M/IO#, CLD/INTA#, HLDA	0
D0 tới D15	Trở kháng cao

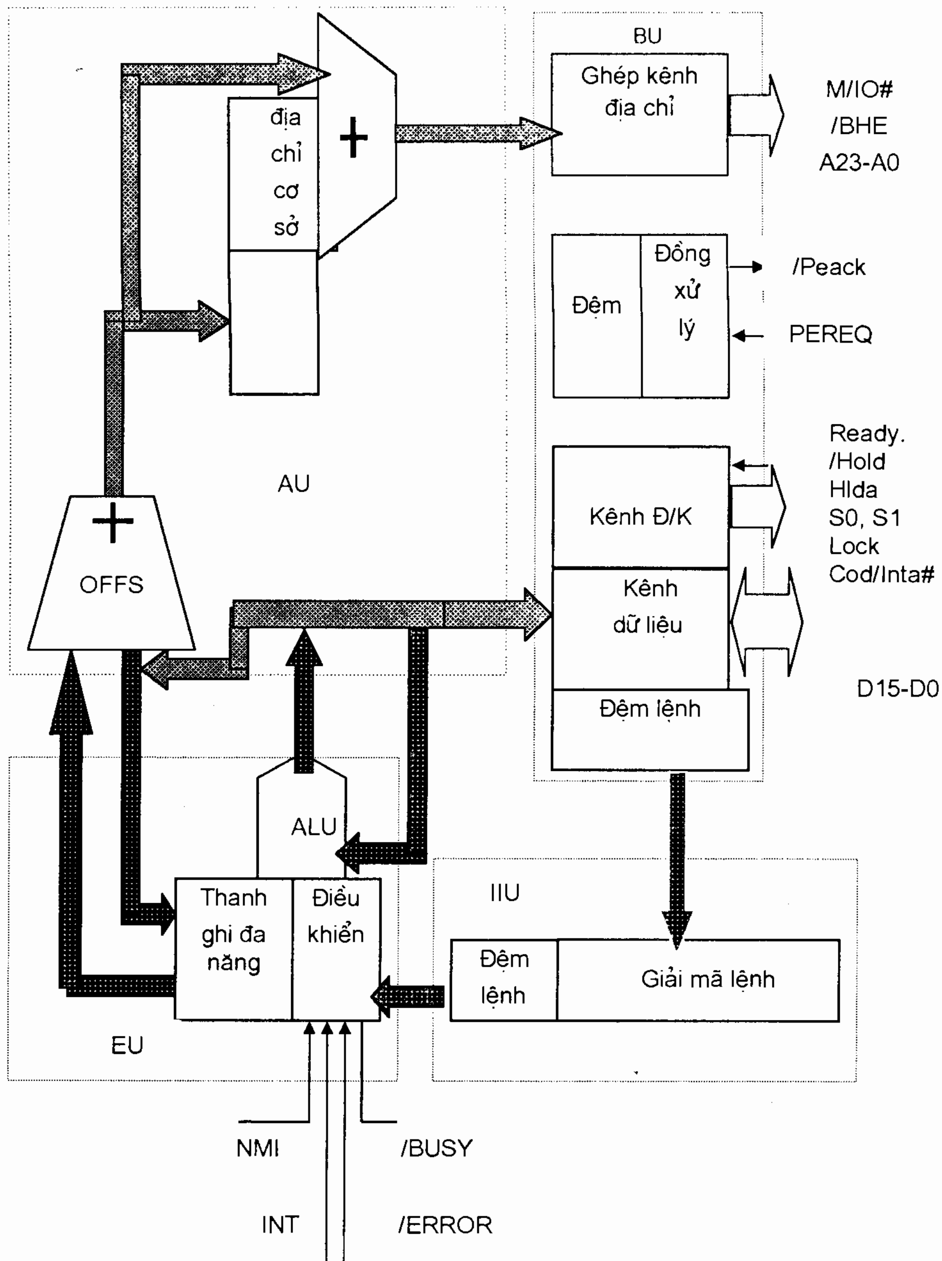
Bộ vi xử lý 80286 được cấu tạo từ 4 đơn vị có thể làm việc song song (hình 2.2). Đó là các đơn vị: BU (BUS UNIT), IU (INSTRUCTION UNIT), EU (EXECUTE UNIT) và AU (ADDRESSED UNIT).

Đơn vị BU phát sinh các tín hiệu địa chỉ, dữ liệu và thông tin điều khiển để thâm nhập vào bộ nhớ hay vào các cổng vào/ra. Nó cho phép thiết lập các quan hệ với các bộ đồng xử lý hoặc với các bộ xử lý đang làm chủ kênh. Đơn vị này cho phép quá trình nhận lệnh diễn ra song song với các quá trình khác nhờ có tệp đệm 6 byte. Nó cho phép loại trừ thời gian chết khi nhận lệnh từ bộ nhớ.

Đơn vị lệnh IU nhận lệnh từ tệp đệm, giải mã rồi đưa vào tệp đợi. Tệp đợi chứa được 3 lệnh.

Đơn vị thực hiện EU sẽ thực hiện lệnh đã được giải mã trong tệp đợi. Nó liên hệ nối bộ nhớ và cổng vào/ra thông qua đơn vị BU.

Đơn vị địa chỉ AU bảo đảm việc bảo vệ, quản lý bộ nhớ và chuyển địa chỉ logic (địa chỉ ảo) thành địa chỉ vật lý (địa chỉ thực) cho đơn vị điều khiển kênh BU.



Hình 2.2. **Tổ chức phần cứng của bộ vi xử lý 80286**

2.1.2. Các thanh ghi của bộ vi xử lý 80286

Bộ vi xử lý 80286 có 15 thanh ghi 16 bit được chia thành 3 nhóm như biểu diễn trên hình 2.3. Nhóm thứ nhất là các thanh ghi đa năng dùng trong các phép tính số học - lôgíc. Bốn thanh ghi 16 bit là AX, BX, CX và DX cũng có thể sử dụng như 8 thanh ghi 8 bit. Ngoài ra chúng còn được chuyên dụng hoá theo chức năng của lệnh, cụ thể:

AX, DX dùng trong các lệnh nhân, chia và vào/ra dữ liệu.

CX dùng trong các lệnh quay vòng, lệnh dịch chuyển và các lệnh lặp.

BX và BP là những thanh ghi cơ sở dùng để chứa địa chỉ cơ sở của một cấu trúc dữ liệu, ví dụ địa chỉ cơ sở của một bảng dữ liệu.

SI, DI là các thanh ghi chỉ số chứa các chỉ số có thể tăng dần khi thâm nhập vào một cấu trúc dữ liệu.

SP là con trỏ ngăn xếp.

Nhóm thứ hai là nhóm các thanh ghi quản lý mảng (đoạn). Các chương trình được cấu trúc từ các môdul lệnh và dữ liệu. Bộ vi xử lý 80286 cho phép một chương trình đang thực hiện có thể thâm nhập cùng một lúc tới 4 mảng nhớ. Có bốn thanh ghi mảng dùng để nhận biết bốn mảng đang sử dụng. Các thanh ghi mảng đó là:

CS là thanh ghi mảng lệnh (COMMAND SEGMENT).

DS là thanh ghi mảng dữ liệu (DATA SEGMENT).

SS là thanh ghi mảng ngăn xếp (STACK SEGMENT).

ES là thanh ghi mảng dữ liệu mở rộng (EXTRA SEGMENT).

Các thanh ghi mảng đôi khi còn được gọi là bộ chọn mảng (chọn đoạn). Hình 2.4 là mô tả phương thức quản lý không gian nhớ theo mảng của bộ vi xử lý 80286.

Nhóm thứ ba là nhóm lệnh điều khiển và trạng thái. Các thanh ghi này bao gồm:

F là thanh ghi cờ có 16 bit được biểu diễn trên hình 2.3.c. CF là cờ nhớ khi được lập sẽ chỉ ra phép nhớ từ bit cao nhất của toán tử 8 bit hay 16 bit. PF là cờ kiểm tra chẵn lẻ. AF là cờ nhớ phụ, phục vụ cho phép tính với mã BCD. Cờ AF phụ thuộc vào thanh ghi AL. ZF là cờ rỗng. SF là cờ dấu. Nếu phép tính có kết quả dương thì SF = 0 còn có kết quả âm thì SF = 1. OF là cờ tràn được dùng trong các phép tính với các toán hạng có dấu.

TF là cờ bẫy cho phép 80286 chạy từng lệnh để giúp cho việc hiệu chỉnh, gỡ rối chương trình. Cờ TF chỉ có thể thay đổi thông qua ngăn xếp: cất thanh ghi cờ F vào ngăn xếp, thiết lập bit TF trong ngăn xếp và đưa trả lại vào thanh ghi cờ. IF là cờ cho phép ngắt. nếu IF = 1 nó sẽ cho phép ghi nhận ngắt ở chân INTR. Lệnh STI sẽ lập cờ IF và lệnh CLI sẽ xoá cờ IF.

DF là cờ hướng cho biết chiều hướng phát triển trong một chuỗi dữ liệu. Nếu DF = 1, nội dung của thanh ghi SI và DI sẽ tự động giảm đi một đơn vị. Nếu DF= 0, nội dung của các thanh ghi trên tự động tăng thêm một đơn vị. Bit DF được lập bằng lệnh STD và được xoá bằng lệnh CLD.

NT (NESTED TASK) chỉ thị rằng nhiệm vụ đang lồng nhau. NT dùng trong chế độ địa chỉ ảo của 80286. IOLP là cờ chỉ mức đặc quyền cửa vào/ra. Nó được dùng trong chế độ bảo vệ bộ nhớ (chế độ địa chỉ ảo).

Con trỏ lệnh IP luôn luôn trỏ tới địa chỉ tương đối (so với địa chỉ đầu của mảng nhớ) của lệnh sẽ thực hiện tiếp theo. Như vậy con trỏ lệnh cùng với thanh ghi mảng lệnh (CS:IP) sẽ định nghĩa thanh đếm lệnh 32 bit (PC). Con trỏ lệnh IP được điều khiển bằng cơ chế ngắt, cơ chế bẫy và các phép nhảy chuyển điều khiển. Hình 2.5 là biểu diễn cách tính địa chỉ thực từ địa chỉ mảng và địa chỉ OFFSET.

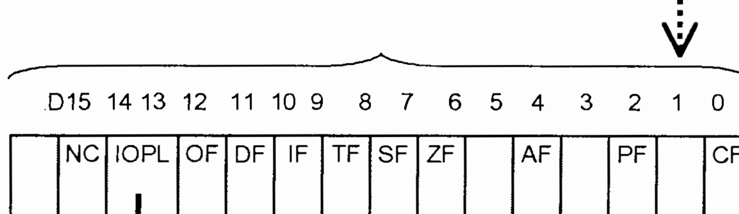
AX	AH	AL	nhân chia,vào/ra dữ liệu
DX	DH	DL	nhân chia,vào/ra dữ liệu
CX	CH	CL	dịch chuyển,quay vòng
BX	BH	BL	TG cơ sở
	BP		TG cơ sở
	SI		TG chỉ số
	DI		TG chỉ số
	SP		con trỏ ngăn xếp

a) Các thanh ghi đa năng

CS	TG chọn mảng lệnh
DS	TG chọn mảng dữ liệu
SS	TG chọn mảng ngăn xếp
ES	TG chọn mảng dữ liệu phụ

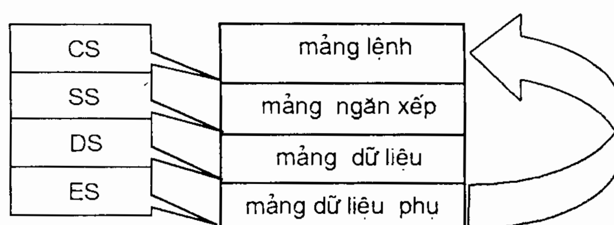
b) Các thanh ghi quản lý mảng

Flag	TG cờ trạng thái
IP	TG con trỏ lệnh (offset)
PSW (trạng thái CPU)	TG trạng thái hệ thống



c) Các thanh ghi điều khiển trạng thái

Hình 2.3. **Các thanh ghi của bộ vi xử lý 80286**

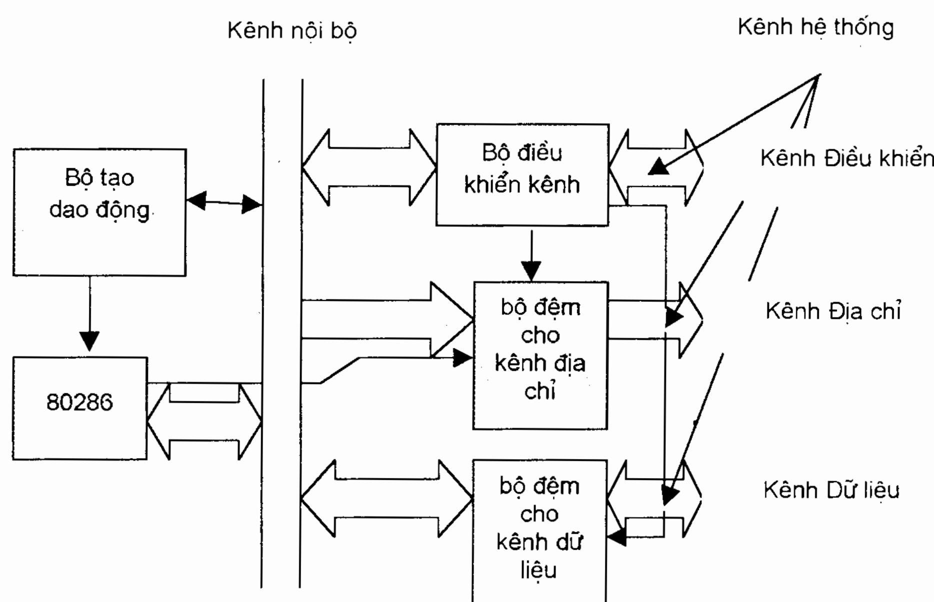


Hình 2.4. **Tổ chức bộ nhớ theo đơn vị mảng**

Thanh ghi trạng thái máy MSW có 16 bit nhưng chỉ có 4 bit được dùng còn các bit khác dự trữ cho bộ vi xử lý 80386 trở lên. PE (PROTECTED MODE ENABLE) cho phép thực hiện chế độ bảo vệ. PE = 1 chỉ ra rằng 80286 đang thực hiện chế độ bảo vệ và chỉ có RESET mới có thể xoá PE về 0. MP (MONITOR PROCESSOR EXTENSION) chỉ ra rằng bộ đồng xử lý cùng làm việc với bộ vi xử lý 80286. EP (EMULATE PROCESSOR EXTENSION) là tín hiệu cho phép mô phỏng một bộ đồng xử lý. TS (TASK SET) là bit thông báo việc chuyển nhiệm vụ và được dùng trong trường hợp có bộ đồng xử lý cùng làm việc.

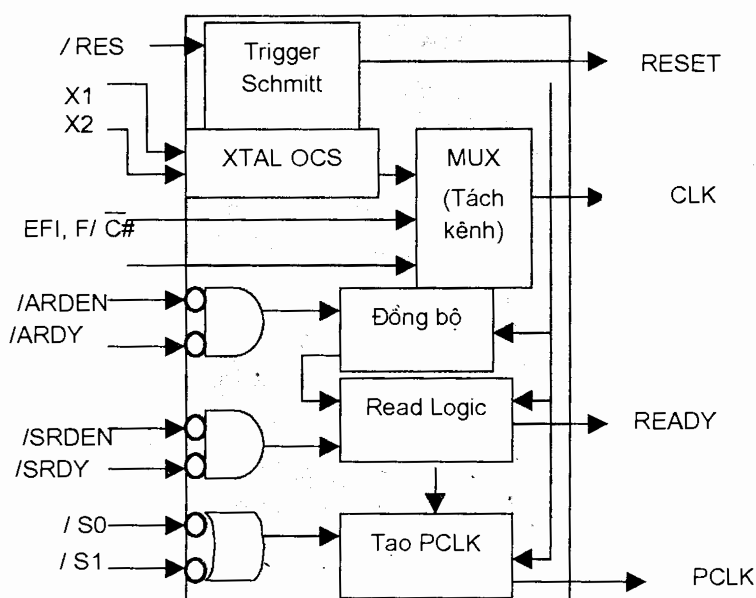
2.3. HOẠT ĐỘNG CỦA BỘ VI XỬ LÝ 80286

Hình 2.5 mô tả kênh hệ thống của một hệ vi xử lý xây dựng trên bộ vi xử lý 80286. Thành phần thứ nhất là bộ đệm cho kênh địa chỉ cục bộ với kênh địa chỉ hệ thống. Thành phần thứ hai là bộ đệm cho kênh dữ liệu nội bộ với kênh dữ liệu hệ thống còn thành phần thứ ba chế biến các tín hiệu điều khiển của kênh cục bộ với kênh hệ thống và một tạo ra xung đồng bộ cho 80286.



Hình 2.5. **Tổ chức kênh hệ thống cho bộ vi xử lý 80286**

Căn cứ vào với sơ đồ khối 2.5, sơ đồ nguyên lý thể hiện trên hình 2.8 có sử dụng các chip IC bổ trợ là bộ tạo dao động 82284 (hình 2.6), Bộ điều khiển kênh (hình 2.7), mạch đệm địa chỉ 74AS533 (3 chip) và mạch đệm dữ liệu 74AS640 (2 chip). Với cách ghép nối như vậy, kênh hệ thống gồm kênh địa chỉ 24 bit, kênh dữ liệu 16 bit và kênh điều khiển với mọi tín hiệu giao tiếp cần thiết đã được hình thành.

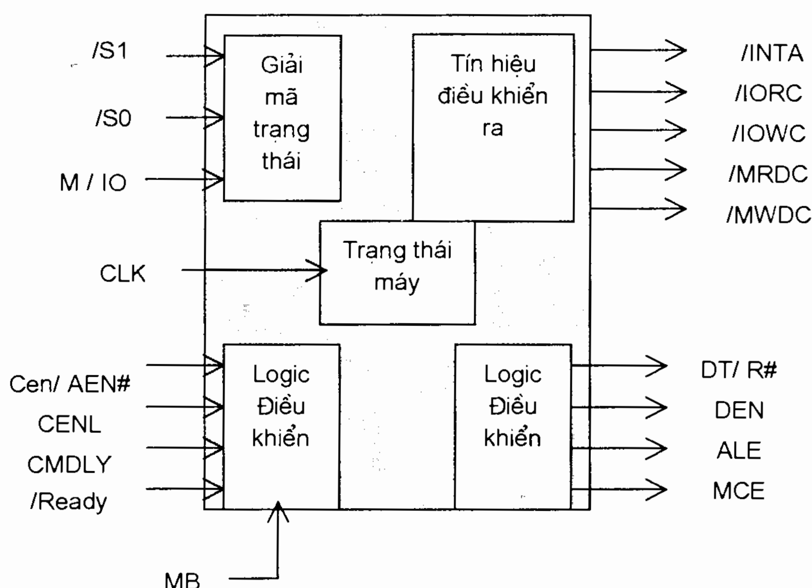


Hình 2.6. **Bộ tạo dao động 82284**

Các chân tín hiệu của bộ tạo giao động 82284 được liệt kê trong bảng 2.4.

Bảng 2.4

CLK	Nhịp hệ thống
/S1/S0	Trạng thái hệ thống
$F/\overline{C}$ [I]	Frecquency/Crystal Select
X1X2 [I]	Đầu vào của phần tử thạch anh
EFI [I]	External Frecquency In
PCLK [O]	Peripheral Clock
$\overline{ARDEN}$ [I]	Asynchronous Ready Enable
$\overline{ARDY}$ [I]	Asynchronous Ready
$\overline{SRDEN}$ [I]	Synchronous Ready Enable
$\overline{SRDY}$ [I]	Synchronous Ready
READY [O]	Ready



Hình 2.7. **Bộ điều khiển kênh 82288**

Các chân tín hiệu của bộ điều khiển kênh 82288 được liệt kê trong bảng 2.5.

Các trạng thái chuyển tiếp của 80286 khi hoạt động bao gồm:

- T_i là trạng thái không tích cực (IDLE).
- T_s là các trạng thái phát các tín hiệu mã chu kỳ máy(STATUS).
- T_c là trạng thái thực hiện lệnh (COMMAND).
- T_h là trạng thái treo của kênh (HOLD).

Bảng 2.5

CLK	Nhip hệ thống
/S1/S0	Trạng thái hệ thống
M/IO#	Chọn bộ nhớ hoặc cổng
MB	MultiBus mode select =1→MultiBus,=0→NonMultiBus
CENL	Command Enable Latched
CMDLY	Command Delay
/READY	
CEN/AEN#	Command Enable / Address Enable. Nếu MB=1 thì là AEN, ngược lại là CEN
ALE	Address Latched Enable
MCE	Master Cascade Enable cho phép bộ điều khiển ngắt chính 8259 chốt địa chỉ
DEN	Data Enable
DT/ R#	Data Transmit/ Receve
IOWC, IORC, MWTC, MRDC	IO Read(Write) Command Memory Read(Write) Command
/INTA	Interrupt Acknowledge

Hình 2.8. **Phương án tổ chức kênh hệ thống của bộ vi xử lý 80286**

Chúng ta xem xét hoạt động cơ sở của bộ vi xử lý 80286 thông qua đồ thị thời gian của chu kỳ máy đọc thông tin (hình 2.9) và chu kỳ máy ghi thông tin (hình 2.10).

Chu kỳ máy đọc thông tin có thể chia làm 6 pha: A, B, C, D, E, F.

Tại pha A (φ2 của T_c thuộc chu kỳ trước) bộ vi xử lý 80286 đưa ra kênh tín hiệu địa chỉ A_{23}- A_0 và các tín hiệu M/IO#, COD/INTA# để thông báo là chu kỳ máy quy chiếu bộ nhớ hay quy chiếu cổng vào/ra.

Tại pha B (φ1 của T_s thuộc chu kỳ đọc) tín hiệu /S1/S0= 01 chỉ ra mã của chu kỳ máy đọc. Tại pha này tín hiệu /BHE cũng có hiệu lực, nếu sử dụng 8 bít cao của kênh dữ liệu D_{15}-D_8.

Pha C (φ2 của T_s thuộc chu kỳ đọc) sẽ đưa tín hiệu ALE ra để chốt các tín hiệu địa chỉ vào mạch 74533 (74373).

Tại pha D (φ1 của T_c thuộc chu kỳ đọc) các tín hiệu /MRDC và DT/R# được đưa về mức tích cực âm. Tín hiệu DEN được đưa về mức tích cực để chọn các mạch truyền dữ liệu. Các tín hiệu trạng thái $/S_0/S_1$ chuẩn bị cho chu kỳ tiếp theo.

Tại pha E (φ2 của T_c thuộc chu kỳ đọc) các tín hiệu M/IO# và COD/INTA# được thiết lập cho chu kỳ máy tiếp theo. Nếu chu kỳ mới là chu kỳ trả lời ngắt hay chu kỳ treo kênh thì các tín hiệu này ở phần trạng thái trở kháng cao.

Cuối pha E, bộ vi xử lý 80286 kiểm tra tín hiệu /READY. Nếu /READY= 0 thì chu kỳ máy đọc thông tin sẽ kết thúc. Dữ liệu đã sẵn sàng trên kênh dữ liệu và bộ vi xử lý 80286 đọc dữ liệu này. Nếu /READY= 1, bộ vi xử lý 80286 đưa ra một trạng thái T_c nữa và cứ như thế đến khi /READY= 0.

Tại pha F (φ1 của T_s của chu kỳ tiếp theo) chu kỳ đọc kết thúc, các tín hiệu DEN, /MRDC, DT/R# trở về trạng thái không tích cực. Mạch 82284 đưa tín hiệu /READY về mức cao.

Chu kỳ ghi thông tin của 80286 bao gồm các pha A, B, C, D, E, F và G.

Tại pha A (φ2 của T_c của chu kỳ trước) bộ vi xử lý 80286 đưa ra kênh địa chỉ A_{23}-A_0, các tín hiệu M/IO và COD/INTA. Quá trình giải mã địa chỉ có thể bắt đầu.

Tại pha B (φ1 của T_s thuộc chu kỳ đang thực hiện) các tín hiệu $/S_1/S_0$ = 00 chỉ ra rằng chu kỳ máy ghi thông tin đang được thực hiện. Tín hiệu sẽ có giá trị 0 hay 1 tuỳ theo các đường dữ liệu là 8 bít cao hay không.

Tại pha C (φ2 của T_s thuộc chu kỳ đang thực hiện) bộ vi xử lý 80286 truyền dữ liệu ra kênh. Tín hiệu ALE chốt địa chỉ còn DEN chọn các mạch truyền dữ liệu. Vì dữ liệu đi từ kênh cục bộ sang kênh hệ thống nên tín hiệu DT/R# = 1.

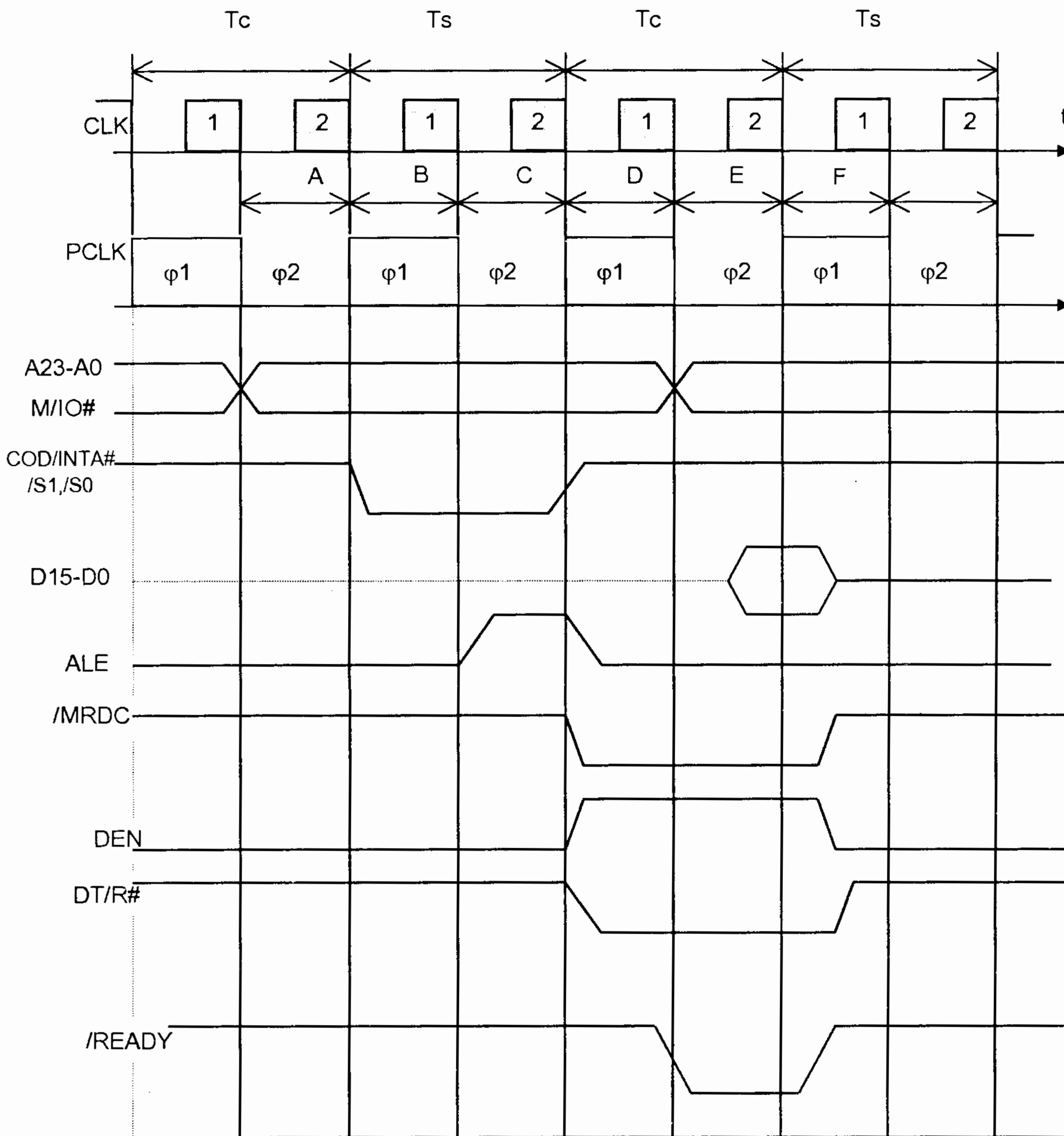
Tại pha D (φ1 của T_c thuộc chu kỳ đang thực hiện), 80286 đưa các tín hiệu /S1/S0 về logíc 1, ALE về lôgíc 0 còn tín hiệu /MWTE có hiệu lực.

Tại pha E (φ2 của T_c thuộc chu kỳ đang thực hiện) các tín hiệu địa chỉ, M/IO# và COD/INTA# của chu kỳ tiếp theo được đưa ra giải mã. Như vậy, chu kỳ máy mới đang được tiến hành ngay khi chu kỳ máy cũ còn đang thực hiện để hoàn tất lệnh. Cuối pha E tín hiệu /READY được trích mẫu, nếu /READY = 0,

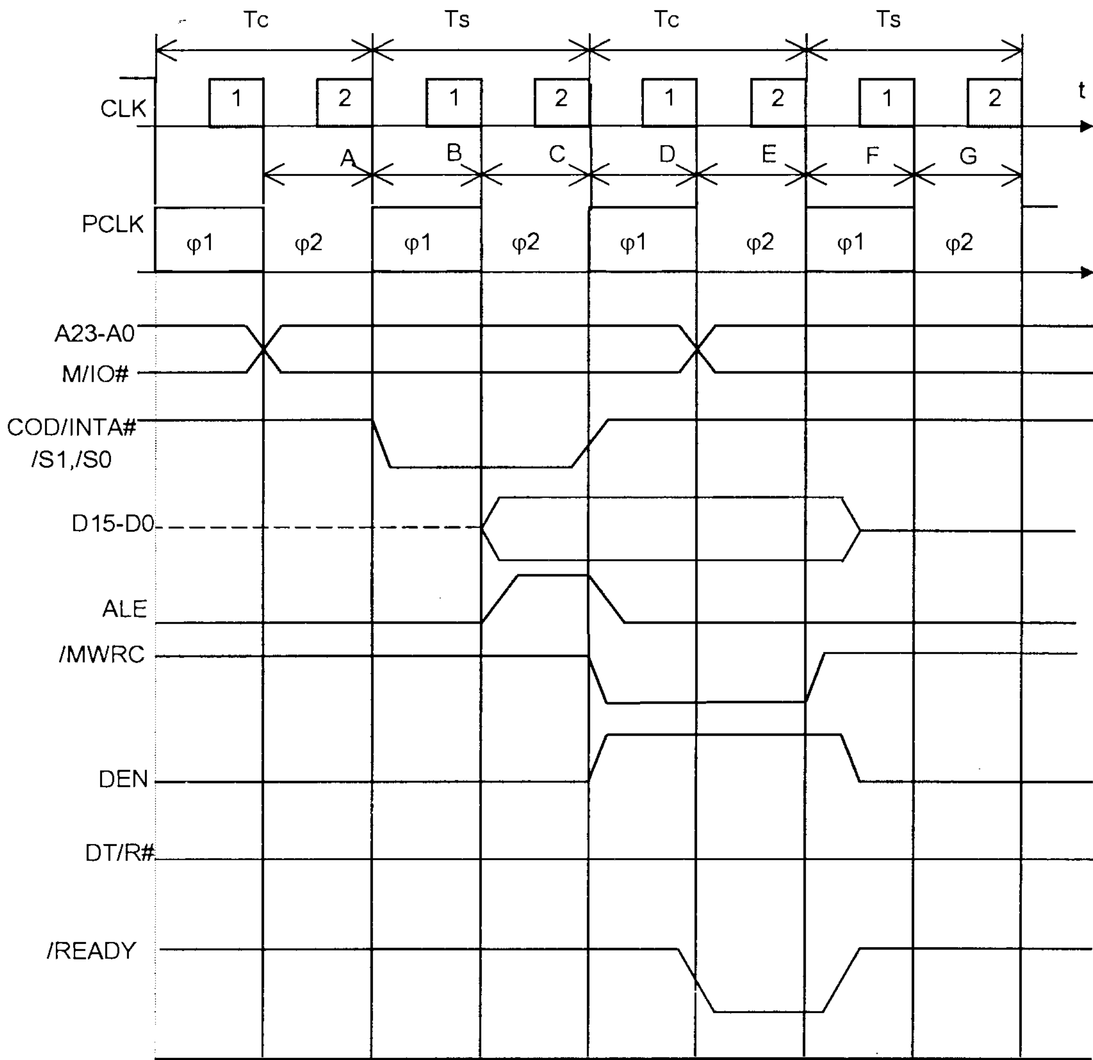
chu kỳ ghi thông tin sẽ kết thúc, ngược lại thì một trạng thái T_c sẽ lại bắt đầu và cứ tiếp tục cho đến khi /READY = 0.

Tại pha F (φ1 của T_s thuộc chu kỳ tiếp theo) các tín hiệu /MWTC và /READY được đưa về mức không tích cực.

Tại pha G (φ2 của T_s thuộc chu kỳ tiếp theo) dữ liệu được đưa vào bộ nhớ, tín hiệu DEN trở về mức thụ động và do đó kênh dữ liệu được giải phóng.



Hình 2.9. **Chu kỳ đọc thông tin của 80286**



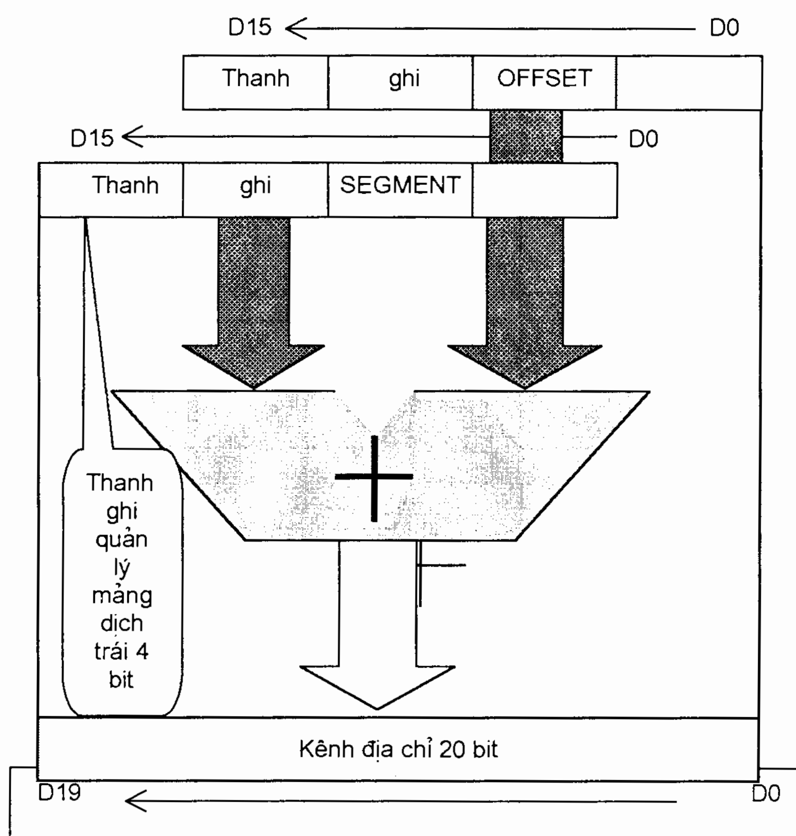
Hình 2.10. **Chu kỳ ghi thông tin của 80286**

2.3. QUẢN LÝ BỘ NHỚ THỰC CỦA BỘ VI XỬ LÝ 80286

2.3.1. Bộ nhớ thực của bộ vi xử lý 80286

Không gian nhớ trong chế độ địa chỉ thực là không gian nhớ mà bộ vi xử lý có thể quản lý trực tiếp bằng giá trị con trỏ do cặp thanh ghi mảng và thanh ghi OFFSET quy định (***SEG : OFFSET***). Không gian nhớ cực đại sẽ phụ thuộc vào số bit của kênh địa chỉ mà bộ vi xử lý dành cho chế độ này. Thí dụ, bộ vi xử lý 80286 sử dụng 20 bit địa chỉ A19-A0 dành cho chế độ địa chỉ thực thì không gian nhớ cực đại sẽ bằng 2^{20} byte tức bằng 1MB. Nếu hệ vi xử lý có IC nhớ (ROM, RAM) nằm ra ngoài không gian này thì bộ vi xử lý 80286 không thể quản lý được.

Trong quá trình hoạt động, khối quản lý bộ nhớ của bộ vi xử lý 80286 cho phép chuyển các giá trị con trỏ do cặp thanh ghi mảng và thanh ghi OFFSET thành những địa chỉ thực và đưa ra kênh hệ thống để kích hoạt các đối tượng tương ứng như bộ nhớ trung tâm hay các cổng vào/ra.



Hình 2.11. **Tạo địa chỉ thực từ nội dung thanh**

2.3.2. Phương pháp địa chỉ hoá của bộ vi xử lý 80286

Bộ vi xử lý 80286 có 6 phương pháp địa chỉ hoá:

- ***Địa chỉ trực tiếp*** cho phép thâm nhập trực tiếp vào toán hạng có địa chỉ là giá trị của SEG:OFFSET.
- ***Địa chỉ gián tiếp*** cho phép thâm nhập trực tiếp vào toán hạng có địa chỉ là giá trị của SEG:(SI hoặc DI hoặc BX).
- ***Địa chỉ tương đối*** cho bởi giá trị của OFFSET cho phép thâm nhập vào toán hạng có địa chỉ là giá trị của tổng giá trị dịch chuyển chứa trong lệnh và nội dung của các thanh ghi cơ sở BX và BP.

OFFSET= (BX hay BP) + Dịch chuyển.

- ***Địa chỉ hoá theo chỉ số*** cho bởi OFFSET cho phép thâm nhập vào toán hạng có địa chỉ là giá trị của tổng giá trị dịch chuuyển chứa trong lệnh và nội dung của thanh ghi SI hay DI.

OFFSET = (SI hay DI) + Dịch chuyển

- ***Địa chỉ tương đối theo chỉ số*** cho bởi OFFSET cho phép thâm nhập vào toán hạng có địa chỉ là giá trị của tổng nội dung thanh ghi cơ sở và nội dung thanh ghi chỉ số.

OFFSET= (BX hay BP) + (SI hay DI)

Khi ta có một vùng dữ liệu động và muốn làm việc với các phần tử của vùng thì chế độ địa chỉ này là thích hợp. Thanh ghi cơ sở phục vụ chọn vùng, còn thanh ghi chỉ số trỏ đến bên trong của vùng xác định.

- ***Địa chỉ tương đối theo chỉ số và giá trị dịch chuyển*** cho bởi OFFSET cho phép thâm nhập đến toán hạng có địa chỉ là giá trị là tổng nội dung của thanh ghi cơ sở, nội dung của thanh ghi chỉ số và giá trị dịch chuyển chứa trong lệnh.

OFFSET = (BX hay BP) + (SI hay DI) + dịch chuyển

Nhờ có thanh ghi cơ sở, chế độ địa chỉ này cho phép lặp lại một cấu trúc mà ở trong đó có các vùng dữ liệu cần tìm. Chỉ số cho phép chọn phần tử xác định bên trong vùng dữ liệu này.

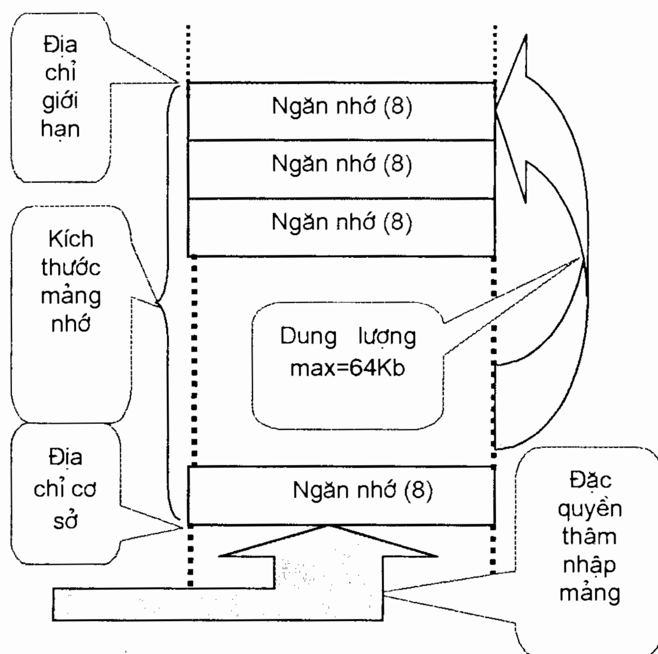
2.4. QUẢN LÝ BỘ NHỚ ẢO CỦA BỘ VI XỬ LÝ 80286

Không gian nhớ trong chế độ địa chỉ ảo là không gian nhớ mà bộ vi xử lý có thể quản lý vượt dung lượng không gian nhớ thực rất nhiều. Khả năng này có được là do tổ chức phần cứng bên trong của bộ vi xử lý được cài đặt một cơ cấu đặc biệt đó là phần ẩn của các thanh ghi quản lý.

Khối quản lý bộ nhớ của bộ vi xử lý 80286 cho phép chuyển các giá trị địa chỉ ảo (hay còn gọi là địa chỉ lôgíc) thành những địa chỉ thực cho bộ nhớ trung tâm.

Nguyên tắc cơ bản của chế độ địa chỉ ảo là phương thức tạo mảng nhớ. Một mảng nhớ được định nghĩa như là một tập hợp các ngăn nhớ liên tiếp có dung lượng không quá 64 KB và mảng này có thể trao đổi giữa bộ nhớ trung tâm và bộ nhớ ngoài.

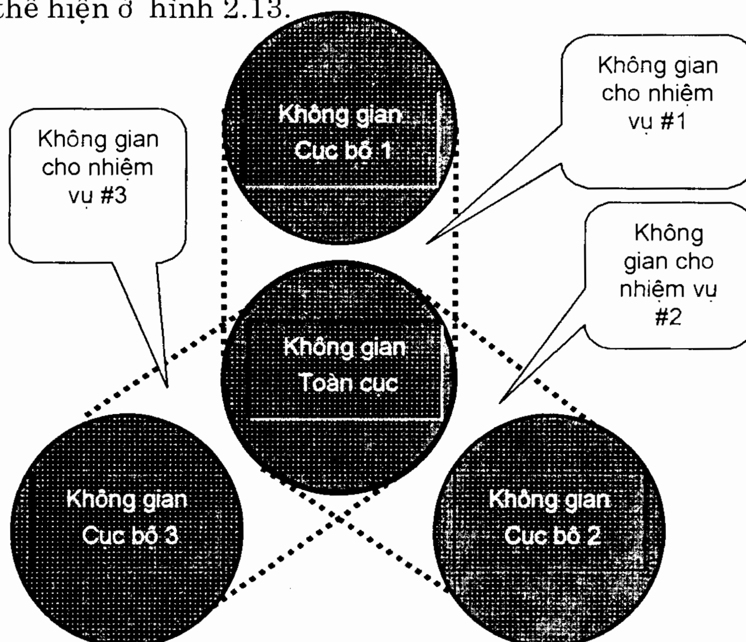
Mỗi mảng nhớ được xác định từ ba tham số: địa chỉ cơ sở, kích thước mảng dung lượng của mảng nhớ và đặc quyền thâm nhập. Mỗi mảng nhớ có cấu trúc như được biểu diễn trên hình 2.12.



Hình 2.12. **Mô tả mảng nhớ trong chế độ địa chỉ ảo**

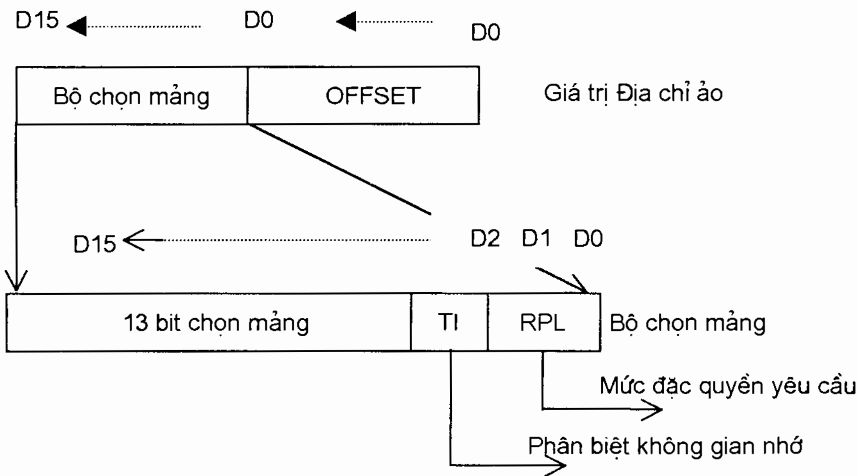
Nhiệm vụ trong phương thức quản lý địa chỉ ảo được hiểu là việc thực hiện một tập hợp các tiến trình gắn với một trạng thái xác định của bộ vi xử lý.

Không gian nhớ luôn luôn gắn với nhiệm vụ. Không gian nhớ được dành riêng cho một nhiệm vụ gọi là không gian nhớ cục bộ. Không gian nhớ mà tất cả các nhiệm vụ đều có thể thâm nhập tới gọi là không gian nhớ toàn cục. nguyên lý này được thể hiện ở hình 2.13.



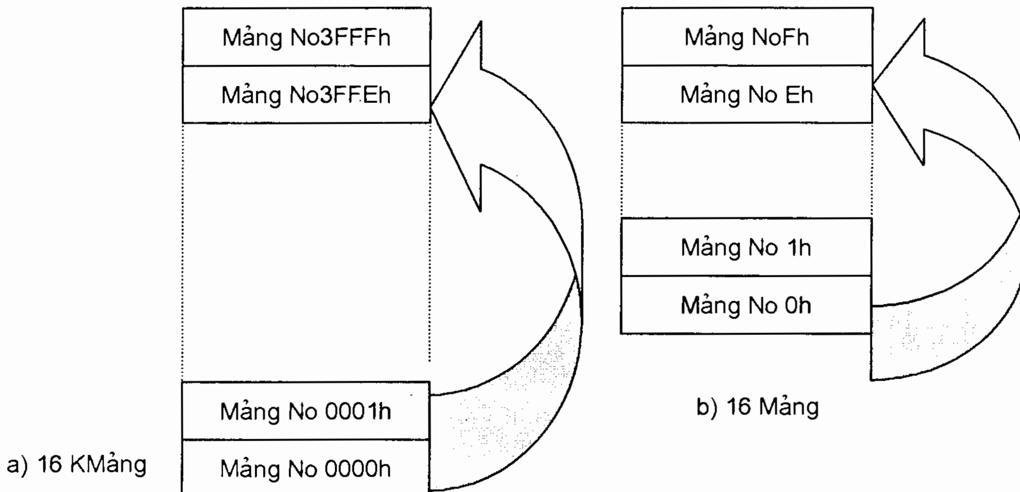
Hình 2.13. **Không gian nhớ trong chế độ địa chỉ ảo**

Một địa chỉ trong chương trình chạy trên vi xử lý 80286 gồm có hai thành phần: bộ chọn mảng 16 bit và OFFSET 16 bit. 32 bit địa chỉ này có ý nghĩa khác nhau khi 80286 làm việc ở chế độ thực và chế độ bảo vệ (hình 2.14). Trong chế độ thực, bộ chọn mảng biểu diễn các bit cao của địa chỉ cơ sở của mảng nhớ. Trong chế độ bảo vệ bộ chọn mảng có ý nghĩa như sau: hai bit thấp dùng để thể hiện ***mức đặc quyền*** của các yêu cầu RPL (Reqested Privelege level); bit TI (Table Indicator) được sử dụng để xác định không gian nhớ. Nếu TI = 0, không gian nhớ là ***không gian toàn cục*** còn nếu TI = 1 thì không gian nhớ là ***không gian cục bộ***. Mười ba bit cao của bộ chọn mảng dành cho chỉ số nên nó xác định được 2^{13} = 8192 mảng nhớ trong không gian nhớ toàn cục và 8192 mảng nhớ trong không gian nhớ cục bộ. Như vậy bộ chọn mảng địa chỉ hoá được cho 16384 mảng nhớ khác nhau.



Hình 2.14. **Con trỏ địa chỉ ảo**

Dung lượng lớn nhất của một mảng nhớ là 64 KB nên không gian nhớ ảo dành cho một nhiệm vụ có dung lượng cực đại là $2^{14}.2^{16} = 2^{30}$ = 1GB. Lưu ý là dung lượng bộ nhớ ở chế độ thực chỉ là $2^{4}.2^{16} = 2^{20}$ = 1MB (hình 3.15).



Hình 2.15. **Dung lượng bộ nhớ ở chế độ a) ảo và b) thực**

Chỉ số giữ vai trò con trỏ đến ***bảng các bộ mô tả***. Bảng này thiết lập quan hệ giữa 32 bit địa chỉ ảo và 24 bit địa chỉ thực của bộ vi xử lý. Bộ vi xử lý quản lý hai loại bảng các bộ mô tả là GDT (Global Descriptor table) tức là bảng các bộ mô tả không gian nhớ toàn cục và bảng các bộ mô tả không gian nhớ cục bộ LDT (Local Deseriptor table). Các bộ mô tả của 80286 bao gồm: bộ mô tả mảng dữ liệu, bộ mô tả mảng lệnh, bộ mô tả mảng hệ thống và bộ mô tả mảng các cổng giao tiếp.

Bộ mô tả mảng dữ liệu được sử dụng để quy chiếu tới mảng dữ liệu và mảng STACK (hình 2.16). Tám byte của bộ mô tả chứa các thông tin về mảng: địa chỉ cơ sở, dung lượng (độ dài) của mảng, một byte đặc quyền thâm nhập vào mảng. Lưu ý rằng hai byte đầu dành cho các bộ vi xử lý cấp cao của hãng Intel (80386, 80486, 80586 ...), do đó khi khởi động phải nạp giá trị 0 vào cho chúng.

Byte chứa giá trị quyền thâm nhập có các bit sau:

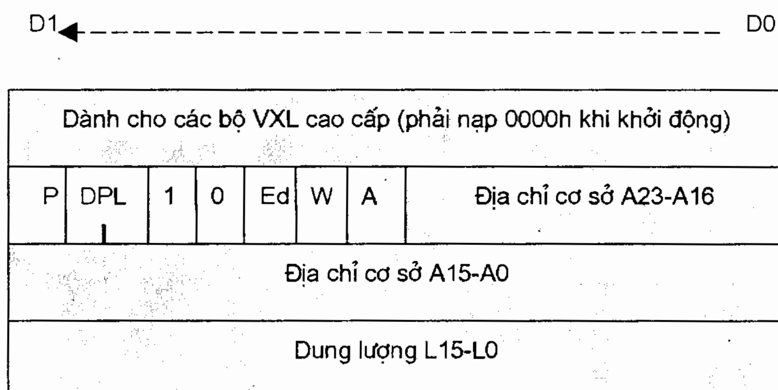
Bit P (Present) = 1 nếu mảng dữ liệu mà bộ mô tả quy chiếu tới đã được nạp trong bộ nhớ, còn nếu chưa thì bit P = 0. Khi chương trình thâm nhập vào mảng dữ liệu chưa có trong bộ nhớ sẽ gây ra ngoại lệ 11 hay 12. Chương trình xử lý ngoại lệ này sẽ nạp mảng dữ liệu cần thiết vào bộ nhớ từ thiết bị ngoại vi (từ ổ đĩa).

Bit DPL (Descriptor Privilege Level) cho biết mức đặc quyền của mảng dữ liệu mà bộ mô tả quy chiếu.

Bit E (Executable) = 0 thông báo rằng bộ mô tả là bộ mô tả mảng dữ liệu.

Bit ED (Expansion Direction) chỉ ra chiều tiến triển của mảng dữ liệu. Nếu ED = 1 thì mảng dữ liệu sẽ thuộc loại STACK. Địa chỉ bắt đầu của mảng sẽ là tổng của địa chỉ cơ sở và độ dài cực đại của mảng, tức là nó phát triển từ vùng địa chỉ cao nhất tới địa chỉ thấp nhất. Nếu ED = 0 thì chiều phát triển của mảng sẽ đi từ địa chỉ thấp nhất tăng dần tới địa chỉ giới hạn.

Bit W (Writable) = 1 thì mảng dữ liệu có thể vừa đọc vừa ghi ra được RW (Read - Write). Nếu W = 0 thì mảng dữ liệu được bảo vệ, tức là cấm ghi, chỉ đọc được thôi, do đó nó có ký hiệu là RO (Read - Only).



Hình 2.16. Bộ mô tả mảng dữ liệu

Bit A (Accesed) = 1 nếu mảng nhớ mà bộ mô tả đã được sử dụng. Một khi A đã được lập thì chỉ có thể xoá nó bằng chương trình. Bit Accesed giúp cho việc thống kê tần xuất thâm nhập vào mảng dữ liệu của một chương trình.

Bộ mô tả mảng lệnh dùng để quy chiếu tới mảng nhớ chứa chương trình. Bộ mô tả lệnh có cấu trúc tương tự như bộ mô tả mảng dữ liệu, riêng byte quyền thâm nhập có một số bít thay đổi như được chỉ ra trên hình 2.17.

D15 ◀-- D0

Dành cho các bộ VXL cao cấp (phải nạp 0000h khi khởi động)							
P	DPL	1	1	C	R	A	Địa chỉ cơ sở A23-A16
Địa chỉ cơ sở A15-A0							
Dung lượng L15-L0							

Hình 2.17. **Bộ mô tả mảng lệnh**

Nếu P = 1 có nghĩa là bộ mô tả quy chiếu tới mảng lệnh.

Nếu R = 0 thì chương trình chứa trong mảng lệnh chỉ có chức năng thực hiện và được ký hiệu là EO (Executable Only) còn nếu R = 1 thì chương trình chứa trong mảng lệnh không những thực hiện được mà còn đọc được nên nó có ký hiệu là ER (Executable and Read).

Nếu C = 0 thì chương trình con được gọi sẽ thực hiện với đặc quyền bằng DPL trong bộ mô tả của mảng chứa chương trình con. Nếu C = 1 thì chương trình con được gọi sẽ thực hiện với đặc quyền bằng DPL trong bộ mô tả quy chiếu mảng chứa chương trình con đó.

Bộ mô tả mảng hệ thống dùng để quy chiếu tới các mảng chứa thông tin cần cho hệ thống (hình 2.18).

D15 ◀-- D0

Dành cho các bộ VXL cao cấp (phải nạp 0000h khi khởi động)							
P	DPL	0	0	0	Kiểu		Địa chỉ cơ sở A23-A16
Địa chỉ cơ sở A15-A0							
Dung lượng L15-L0							

Hình 2.18. **Bộ mô tả mảng hệ thống**

Nếu kiểu = 1, thì bộ mô tả quy chiếu tới mảng chứa trạng thái của nhiệm vụ TSS (Task State Segment). Nhiệm vụ này không ở trạng thái thực hiện. Nếu kiểu = 3, bộ mô tả quy chiếu mảng TSS của một nhiệm vụ đang hoạt động. Nếu kiểu = 2, bộ mô tả quy chiếu mảng chứa bảng các bộ mô tả cục bộ.

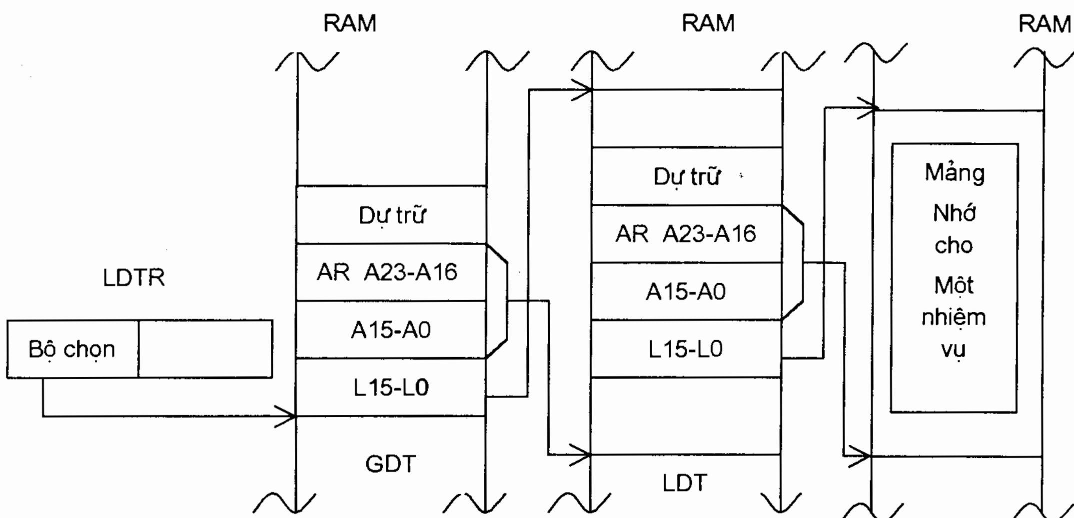
Thông tin trong bảng các bộ mô tả bao gồm thông tin trong GDT và LDT.Trong GDT chứa các bộ mô tả mảng tương ứng với tất cả các mảng nhớ trong không gian nhớ toàn cục, còn trong LDT chứa các bộ mô tả mảng nhớ trong không gian nhớ cục bộ của một nhiệm vụ.

Mỗi bảng các bộ mô tả cũng chính là một mảng nhớ được định nghĩa bằng một bộ mô tả mảng đặc biệt, thuộc nhóm bộ mô tả mảng hệ thống.

GDT là một bảng duy nhất nên không cần xác định trước bằng một bộ mô tả riêng. Địa chỉ và kích thước của mảng GDT được chứa trong một thanh ghi đặc biệt, gọi là thanh ghi bảng các bộ mô tả toàn cục GDTR (Global Descriptor Table register).

LDT được xác định bằng các bộ mô tả ở trong bảng GDT. Thông tin về địa chỉ cơ sở và kích thước của mảng chứa bảng các bộ mô tả cục bộ tương ứng với nhiệm vụ đang thực hiện được chứa trong thanh ghi bảng các bộ mô tả cục bộ LDTR (Local Descriptor Table register). Nội dung của thanh ghi này sẽ thay đổi khi chuyển từ nhiệm vụ này sang nhiệm vụ khác.

Cơ cấu thâm nhập vào một mảng nhớ được thể hiện qua hình 2.19.



Hình 2.19. **Cơ chế thâm nhập vào một mảng nhớ trong chế độ địa chỉ ảo**

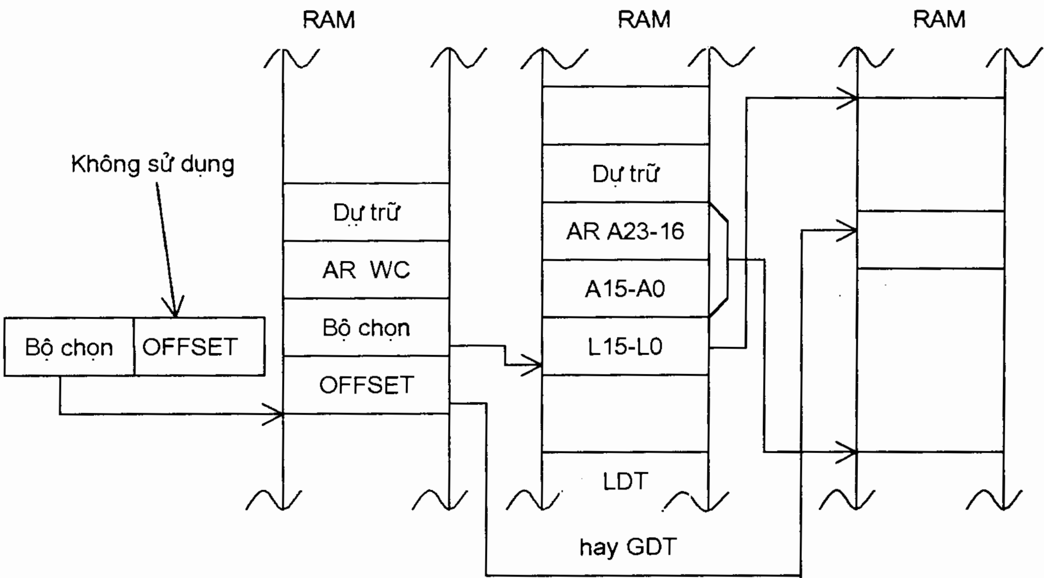
Bộ mô tả các cổng giao tiếp có dạng như hình 2.20. Các lệnh CALL và JMP chỉ có thể thâm nhập vào mảng lệnh có mức đặc quyền cao hơn thông qua một cổng nối ghép gọi là cổng giao tiếp. Có tất cả bốn loại cổng giao tiếp: cổng kiểu gọi (CALL GATE); cổng kiểu bẫy (TRAP GATE); cổng theo nhiệm vụ (TASK GATE).

D15<-- D0

Dành cho các bộ VXL cao cấp (phải nạp 0000h khi khởi động)						
P	DPL	0	0	1	Kiểu	X X X Số từ (WC)
Bộ chọn						X X
OFFSET (không dùng kiểu 01, 11)						

Hình 2.20. **Bộ mô tả mảng giao dịch**

Cơ chế thâm nhập vào một mảng nhớ thông qua cổng giao tiếp được mô tả ở hình 2.21.

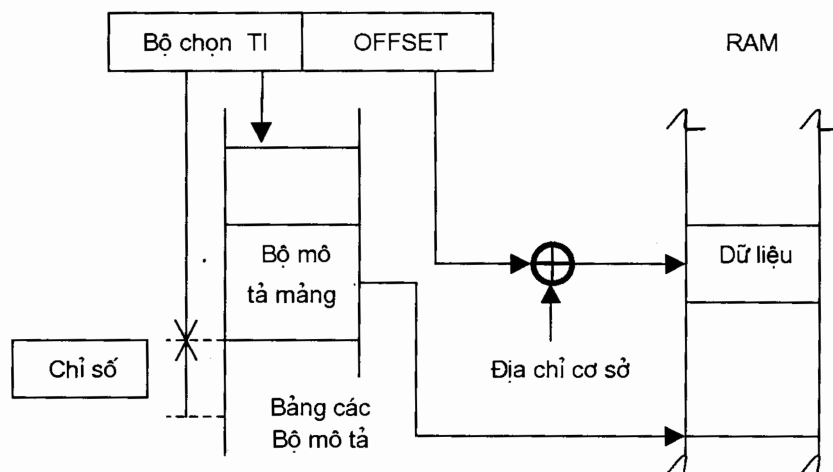


Hình 2.21. **Cơ chế thâm nhập vào một mảng nhớ thông qua cổng giao dịch CALL**

2.5. PHƯƠNG PHÁP TÍNH ĐỊA CHỈ VẬT LÝ (THỰC) TỪ ĐỊA CHỈ ẢO

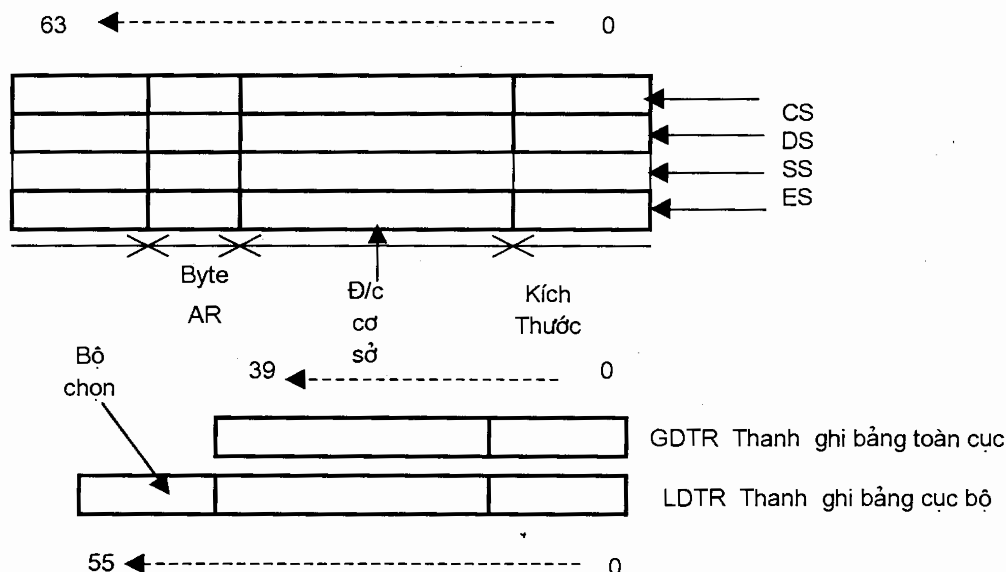
Địa chỉ ảo của 80286 có 32 bit bao gồm 16 bit của bộ chọn và 16 bit OFFSET. Bộ chọn có ba thành phần: chỉ số, TI, và RPL. TI cho biết bộ mô tả

thuộc GDT hay LDT. Vì mỗi bộ mô tả mảng có 8 byte nên địa chỉ của bộ mô tả trong bảng sẽ là địa chỉ cơ sở cộng với chỉ số nhân 8 lần. 80286 sẽ tìm thấy trong bộ mô tả địa chỉ cơ sở của mảng nhớ thực và giới hạn của nó. Cộng 24 bit địa chỉ cơ sở với 16 bít OFFSET trong địa chỉ ảo sẽ cho 24 bit địa chỉ thực của mảng nhớ. Cách tính địa chỉ thực được biểu diễn trên hình 2.22.



Hình 2.22. **Tính địa chỉ vật lý**

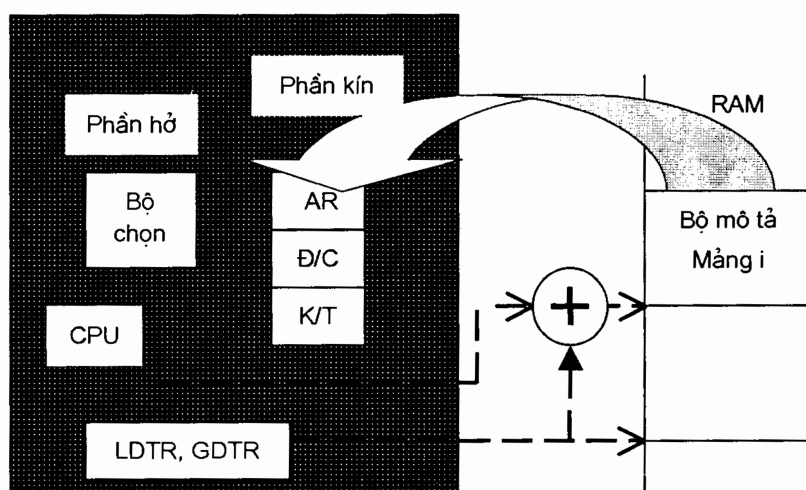
Quản lý bộ nhớ ảo được thực hiện nhờ có các thanh ghi quản lý bộ nhớ đặc biệt. Cấu trúc của các thanh ghi này được biểu diễn trên hình 2.23. Các thanh ghi mảng CS, DS, ES, SS có hai phần: phần hở là bộ chọn 16 bit, phần kín 48 bit bao gồm: một byte thể hiện đặc quyền thâm nhập, ba byte địa chỉ cơ sở của mảng và hai byte kích thước mảng.



Hình 2.23. **Thanh ghi quản lý cơ chế địa chỉ ảo**

Bộ chọn 16 bit được nạp giá trị bằng các lệnh LDS, LES, MOV. Các lệnh này làm thay đổi nội dung của SS, DS và ES. Các lệnh CALL và JMP làm thay đổi nội dung của CS.

Trong khi thực hiện các lệnh này, bộ chọn của địa chỉ logíc được nạp vào phần cao của các thanh ghi. 80286 sử dụng bộ chọn (chỉ số TI) để thâm nhập vào bộ mô tả 48 bit và nó được tự động sao sang phần kín của thanh ghi mảng. Quá trình này được thể hiện qua hình 2.24.



Hình 2.24. **Nguyên tắc thâm nhập vào bộ mô tả và copy vào CPU**

Như vậy, thông qua thanh ghi mảng, bộ vi xử lý 80286 biết tất cả các tính chất của mảng nhớ đang sử dụng. 80286 dùng nội dung của thanh ghi này cùng 16 bit OFFSET của địa chỉ lôgíc thâm nhập vào bên trong mảng, tránh được những tìm kiếm trong các bảng ở bộ nhớ.

Thanh ghi GDTR chứa địa chỉ cơ sở và giới hạn của mảng GDT. Bộ vi xử lý sử dụng các lệnh LGDT (Load) và SGDT (Store) để nạp hoặc cất giữ nội dung của nó.

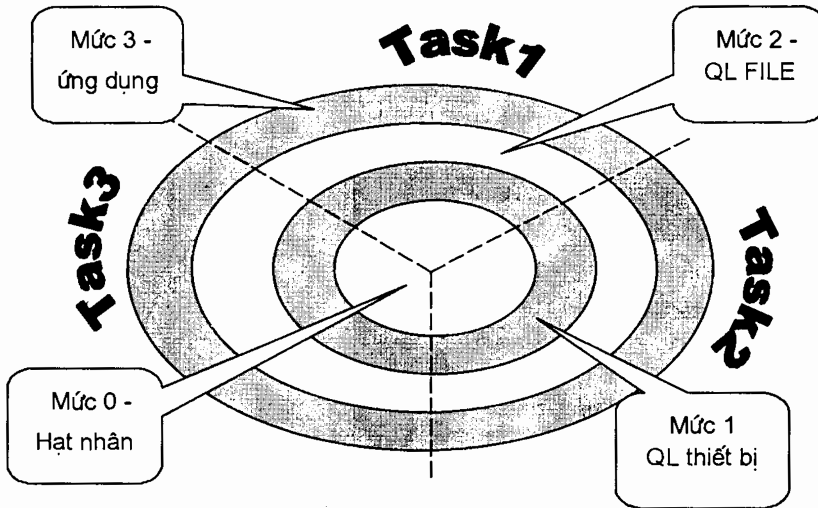
Thanh ghi LDTR có hai thành phần: phần chọn 16 bit, phần kín bao gồm địa chỉ cơ sở của bảng LDT đang dùng và giới hạn của bảng LDT đang dùng.

Bộ chọn được nạp nội dung bằng lệnh LLDT (load). Bộ vi xử lý 80286 sẽ sao các thông tin về địa chỉ cơ sở và giới hạn của bảng LDT vào phần kín của thanh ghi LDTR. Trong quá trình thực hiện, bộ chọn của LDTR sẽ thay đổi mỗi khi chuyển nhiệm vụ. Do vậy ứng với mỗi nhiệm vụ sẽ có một bảng các bộ mô tả cục bộ LDT.

2.6. BẢO VỆ BỘ NHỚ TRONG CHẾ ĐỘ ĐỊA CHỈ ẢO

Bảo vệ bộ nhớ có các chức năng sau: cách ly chương trình hệ thống và chương trình ứng dụng, cách ly giữa các nhiệm vụ và kiểm tra thời điểm thâm nhập vào đối tượng.

Bộ vi xử lý 80286 có bốn mức đặc quyền (hình 2.25), trong đó mức 0 là mức đặc quyền cao nhất còn mức 3 là mức đặc quyền thấp nhất. Mỗi mảng được phân bổ một mức đặc quyền.



Hình 2.25. **Phân cấp mức đặc quyền**

Chương trình được tạo ra từ các mảng lệnh và mảng dữ liệu. Mức đặc quyền phân bố cho một chương trình cho biết chương trình có quyền làm những gì khi nó được thực hiện bởi một nhiệm vụ. Mức đặc quyền của một nhiệm vụ thay đổi theo thời gian và phụ thuộc vào mức đặc quyền của chương trình đang chạy.

Hạt nhân bao gồm các chương trình sơ đẳng quản lý các tài nguyên của bộ vi xử lý và bộ nhớ. Hạt nhân phải gọn, có khả năng vận hành tốt, không bị hỏng do phần mềm của các lớp có mức đặc quyền thấp hơn.

Mức 1 chứa tất cả các miền liên quan tới nhiệm vụ quản lý hệ điều hành như: thiết lập mức ưu tiên giữa các nhiệm vụ, nạp thuật toán trao đổi (Swapping) và quản lý các cổng vào/ra.

Mức 2 bao gồm các chức năng quản lý tệp, quản lý thư viện, đó là các bộ ghép nối mềm cho các chương trình ứng dụng.

Mức 3 dành cho các chương trình ứng dụng.

Nguyên tắc bảo vệ bộ nhớ đòi hỏi mỗi đối tượng (mảng) nhớ phải có bộ mô tả cho biết mức đặc quyền của đối tượng đó. DPL (Descriptor Privilege Level) được mã hoá bằng hai bit (D5D6) của byte quyền thâm nhập trong bộ mô tả mảng. Đối với tất cả các bộ mô tả mảng, đó chính là mức đặc quyền của mảng. CPL (Current Privilege Level) là mức đặc quyền đang thực hiện tại thời điểm cho trước. Đó chính là các bit RPL của bộ chọn mảng lệnh đang chạy. RPL (Riquested Privilege Level) là mức đặc quyền yêu cầu, được thể hiện bằng hai bit (D0D1) của bộ chọn.

Quy tắc đơn giản của cơ chế bảo vệ là:

RPL = DPL (DPL thuộc bộ mô tả được định nghĩa bởi bộ chọn).

EDL (Effective Privilege Level) là số cực đại trong hai số CDL và RPL.

Chương trình đang thực hiện có thể thâm nhập một cách tự do vào các mảnh lệnh và mảng dữ liệu cùng mức đặc quyền với chương trình đó. Khi điều khiển vượt ra ngoài mức đặc quyền của chương trình đang chạy thì phải tuân theo các quy tắc riêng.

Chương trình đang chạy chỉ có thể thâm nhập vào các mảng dữ liệu có mức đặc quyền bằng hay thấp hơn mức đặc quyền của nó, nghĩa là:

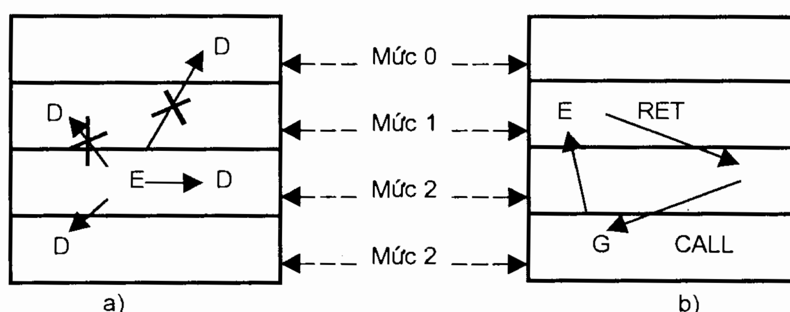
CPL<= DPL

Phép kiểm tra mức đặc quyền sẽ xảy ra khi nạp bộ chọn. Thí dụ khi thực hiện lệnh

MOV DS, AX; AX chứa bộ chọn của mảng dữ liệu.

Quy tắc thâm nhập vào mảng dữ liệu được thể hiện trên hình 2.26a.

Quy tắc cơ bản để gọi một mảng lệnh bằng lệnh CALL hoặc lệnh JMP là: CPL = DPL, trong đó DPL thuộc mảng lệnh đích (hình 2.26 b). Thông qua cửa giao dịch có thể thâm nhập vào mảng lệnh có mức đặc quyền cao hơn mức đặc quyền đang thực hiện, tức là: CPL >= DPL với DPL thuộc mảng lệnh đích. Bất cứ một sự thâm nhập nào trái với quy tắc trên đều sinh ra một ngoại lệ.



Hình 2.26. a) Quy tắc thâm nhập vào mảng dữ liệu, b) Quy tắc thâm nhập vào mảng dữ liệu (D-mảng DATA; E-mảng lệnh; x - cấm)

Các lệnh đặc quyền chỉ có thể thực hiện ở mức đặc quyền 0 (CPL = 0). Các lệnh đặc quyền bao gồm:

LGDT là lệnh nạp vào GDTR địa chỉ cơ sở 24 bit giới hạn 16 bit.

LIDT là lệnh nạp vào IDTR địa chỉ cơ sở 24 bit giới hạn 16 bit.

LLDT là lệnh nạp phần chọn cho LDTR 16 bit.

LTR là lệnh nạp phần chọn cho TR, 16 bit.

LMSW là lệnh nạp từ trạng thái cho hệ.

CLTS là lệnh xoá bit TS.

HALT là lệnh dừng hoạt động của 80286.

Các lệnh POPF và IRET không phải là các lệnh đặc quyền nhưng nó lại có thể thay đổi các bit IOPL của thanh ghi F, khi nó đang được thực hiện ở mức đặc quyền 0.

2.7. KHỞI ĐỘNG BỘ VI XỬ LÝ 80286

Sau khi khởi động, các trạng thái của bộ vi xử lý được xác lập theo bảng 2.6. Cụ thể:

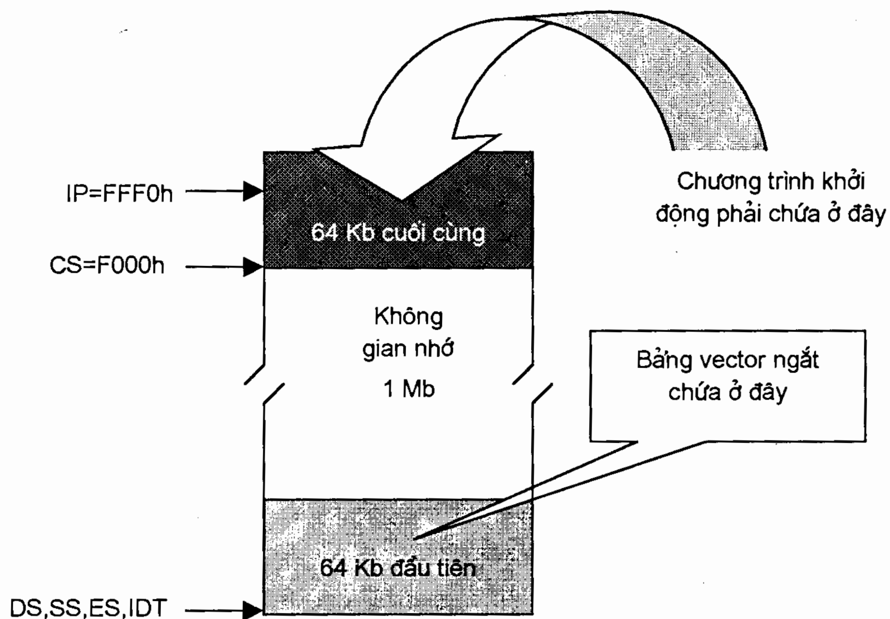
- *Tín hiệu INTR bị che.*
- *Bộ vi xử lý 80286 ở chế độ thực, không làm việc với bộ đồng xử lý (PE = 0, EM = 0, MP = 0).*
- *Bảng các véc tơ ngắt xác định ở địa chỉ 000000.*
- *DS, ES, SS được khởi động để cho phép thâm nhập vào 64 KB đầu của bộ nhớ.*
- *Tổ hợp* ***CS:IP*** *là địa chỉ bắt đầu của chương trình sau RESET:*

  F0000h
  0FFF0h
  ______
  FFFF0h

- *Vùng nhớ chứa chương trình khởi động của hệ thống là 64 KB cuối cùng của không gian nhớ (hình 2.27).*

Bảng 2.6

Các Thanh Ghi	*Giá trị (HEX)*
F	0002
MSW	FFF0
IP	FFF0
Bộ chọn CS	F000
Bộ chọn DS	0000
Bộ chọn SS	0000
Bộ chọn ES	0000
Địa chỉ cơ sở của CS	FF0000
Địa chỉ cơ sở của DS	000000
Địa chỉ cơ sở của SS	000000
Địa chỉ cơ sở của ES	000000
Giới hạn của CS	FFFF
Giới hạn của DS	FFFF
Giới hạn của SS	FFFF
Giới hạn của ES	FFFF



Hình 2.27. **Bố trí bộ nhớ sau khi RESET**

CHƯƠNG 3
LẬP TRÌNH ASSEMBLY CHO HỆ VI XỬ LÝ

Với khả năng của bộ vi xử lý đã trình bầy ở chương 2, chúng ta có đủ điều kiện để thiết kế phần mềm quy định hoạt động chức năng của bất kỳ hệ vi xử lý nào được tổ chức trên bộ vi xử lý 80286 INTEL (và các bộ vi xử lý tương thích cấp cao hơn 16/32 bit của hãng INTEL). Ngôn ngữ lập trình chính để xây dựng phần mềm cho hệ vi xử lý là ngôn ngữ bậc thấp ASSEMBLY vì có các ưu điểm sau:

- Sử dụng trực tiếp tập lệnh của bộ vi xử lý nên quá trình điều hành chức năng rất sát với cấu trúc phần cứng của hệ thống nên triệt để khai thác được khả năng của phần cứng mà không một ngôn ngữ bậc cao nào (Pascal, C..) có thể làm được.
- Có tốc độ thực hiện nhanh nhất so với các ngôn ngữ khác. Do vậy nó rất thích hợp với dạng chức năng có yêu cầu thời gian thực ngặt nghèo như các thao tác với tín hiệu biến đổi nhanh.

3.1. TỔNG QUAN VỀ NGÔN NGỮ ASSEMBLY

Khi hệ vi xử lý thực hiện một chương trình, nó nhận từng lệnh được biểu diễn dưới dạng một dãy số nhị phân (mã binary) bao gồm các giá trị logic 0 và 1, giải mã rồi thực hiện. Các lệnh nhị phân này thường được biểu diễn ở hệ cơ số 16 (mã HexaDecimal viết tắt là mã Hexa). Các lệnh của chương trình được viết bằng mã nhị phân hay mã Hexa được gọi là chương trình viết bằng ngôn ngữ máy.

Khi chương trình được viết trực tiếp bằng ngôn ngữ máy, nếu muốn thêm hoặc xoá một mã lệnh nào đó thì các mã lệnh có địa chỉ đi kèm như jump, call, loop ... cũng phải được tính toán lại cho đúng nên rất khó khăn cho việc viết và sửa chương trình. Vì vậy thay vì viết trực tiếp bằng ngôn ngữ máy, người lập trình có thể viết bằng một ngôn ngữ dưới dạng các ký hiệu hình thức hoặc các từ gợi nhớ tuân theo một qui tắc nào đó dễ đọc và dễ hiểu. Mỗi từ gợi nhớ này tương đương với một lệnh của CPU. Ngôn ngữ này chính là ngôn ngữ ASSEMBLY(assembly language). Chính xác hơn thì ***assembly language*** là

ngôn ngữ lập trình cấp thấp gần với ngôn ngữ máy còn ***assembler*** là chương trình dịch các chương trình viết bằng assembly language sang mã máy cho bộ vi xử lý.

Các chương trình biên dịch thông dụng hiện nay là Macro assembler của hãng Microsoft và Turbo assembler của hãng Borland. Chương trình dịch sẽ nhờ môi trường soạn thảo của một cửa sổ nào đó miễn là nó sử dụng bộ ký tự chuẩn ASCII. Có thể dùng cửa sổ soạn thảo của Turbo Pascal, Turbo C hoặc sử dụng ngay môi trường soạn thảo của Norton Utilies.

3.2. CÁC THÀNH PHẦN CƠ BẢN CỦA ASSEMBLY

3.2.1. File nguồn assembly

File nguồn của assemly gồm một tập hợp các phát biểu hợp ngữ (assembly language statement). Mỗi một phát biểu được viết trên một dòng, nó có thể là một lệnh (assemly language instruction) hay một chỉ dẫn (assembler directive).

Các lệnh assemly giống như các lệnh của CPU, nó xác định các hành động mà CPU sẽ thực hiện. Khi assembler dịch một file nguồn của assemly thì mỗi lệnh File nguồn của assemly sẽ được dịch sang một lệnh mã máy tương ứng.

Các lệnh chỉ dẫn chỉ là lệnh cho riêng chương trình biên dịch assembler mà không phải là lệnh của CPU. Các lệnh này tuy xuất hiện trong chương trình nhưng không được dịch sang mã máy tương ứng. Các chỉ dẫn dùng để điều khiển cách dịch của assembler đối với các lệnh khi dịch một file nguồn sang file mã máy.

3.2.2. Bộ ký tự, từ khóa, tên của assembly

Một ngôn ngữ bất kỳ, từ ngôn ngữ giao tiếp của người đến ngôn ngữ máy tính, đều được xây dựng dựa trên một bộ ký tự và các từ có nghĩa được gọi chung là từ vựng. Tiếp theo là phải có qui tắc viết các từ thành câu để diễn tả các hành động, sự việc cần thực hiện nghĩa là phải tuân theo cú pháp và ngữ pháp của ngôn ngữ đó. Do đó bộ ký tự assembly được xây dựng dựa vào các ký tự sau:

Bộ 26 chữ cái la tinh gồm 26 chữ cái lớn A-Z và 26 chữ cái nhỏ a-z.

Bộ chữ số thập phân : 0, 1, 2,3,4,5,6,7,8,9.

Các ký tự đặc biệt như: ? @ - $. : [ ] ( ) < > { } + - / *

& % ! '\ = #^ ;,' ' "

Các ký tự ngăn cách gồm khoảng trắng và tab.

Từ vựng gồm:

Từ khoá (keyword): là các từ dành riêng của assembly như tên các thanh ghi, tên gợi nhớ của các phát biểu assembly, tên các toán tử ... Các từ khoá này đòi hỏi người lập trình khi dùng phải viết đúng như assembler quy định. Các từ khóa có thể viết bằng chữ hoa hoặc chữ thường, giá trị và ý nghĩa không thay đổi. Tuy nhiên để cho chương trình sáng sủa và rõ ràng chúng ta nên viết các từ khoá bằng chữ hoa.

Tên (Symbol Name): là một dãy ký tự dùng để biểu thị tên hằng tên biến, tên nhãn, tên chương trình con, tên mảng ...

Tên do người lập trình đặt có thể chứa:

Các ký tự chữ cái hoa và thường A - Z và a-z, assembler khi dịch không phân biệt chữ hoa và chữ thường.

Các chữ số 0 - 9

Các ký tự đặc biệt (.), (?), (@), (-), ($)

Ký tự đầu tiên của tên không thể là ký tự số mà phải là ký tự chữ hoặc các ký tự đặc biệt để assembler có thể phân biệt sự khác nhau giữa tên và số.

Không thể đặt tên mới trùng tên với các từ khoá của assembly. Để chương trình sáng sủa tên phải được đặt sao cho có ý nghĩa đối với nhiệm vụ cụ thể.

3.2.3. Cấu trúc một lệnh của assembly

Một lệnh assembly gồm 4 phần:

[nhãn:] tên lệnh [toán hạng] [; ghi chú]

Mỗi dòng chỉ chứa duy nhất một lệnh assembly và mỗi lệnh phải nằm trên một dòng.

Mỗi phần phải cách nhau bằng một hoặc nhiều ký tự ngăn cách.

Các phần có thể có hoặc không tuỳ ý nghĩa câu lệnh.

Mỗi dòng dài tối đa 128 ký tự.

Nhãn (label). Một câu lệnh có thể có một nhãn đứng trước. Nhãn là một tên dùng để thay thế địa chỉ câu lệnh. Điều này cho phép các câu lệnh khác có thể tham chiếu đến câu lệnh có nhãn thông qua nhãn của nó. Một nhãn có thể dài bất kỳ, nhưng chỉ có 31 ký tự đầu được dùng, các ký tự còn lại assembler bỏ qua.

Tên lệnh xác định hành động của câu lệnh. Mỗi tên lệnh gồm từ 2 đến 7 ký tự. Assembler sẽ dùng một bảng (internal table) để chuyển đổi các tên gợi nhớ thành các lệnh mã máy tương ứng mà CPU có thể hiểu được.

Toán hạng (operand) xác định các dữ liệu mà câu lệnh cần xử lý. Các toán hạng là các hằng, thanh ghi, tên tượng trưng, biểu thức tuỳ thuộc vào kiểu địa chỉ được dùng. Các phép địa chỉ hoá trong assembly cũng giống như các phép định địa chỉ trong CPU đã nói ở chương 2. Toán hạng được cấu tạo bởi các đại lượng sau:

Hằng (constant) là giá trị không đổi dùng trong chương trình.

Thanh ghi: Tất cả tên các thanh ghi của CPU đều có thể dùng trong phần toán hạng của một lệnh hợp ngữ. Lưu ý rằng tên thanh ghi của CPU chính là địa chỉ truy cập của nó.

Tên tượng trưng (symbol) là một tên dùng để biểu thị một hằng, một biến, địa chỉ nhãn, đoạn, chương trình con, macro, record và structure.

Biểu thức: (expession) cho ta một công thức tính toán một giá trị theo một qui tắc toán học.

Một biểu thức đơn giản gồm toán tử (operator) và toán hạng (operand). Toán hạng có thể là hằng, thanh ghi hay tên tượng trưng. Toán tử chỉ tác vụ cần thực hiện khi tổ hợp các toán hạng của một biểu thức. Toán tử là các phép toán +, - , *, ?, div, mod ...

Biểu thức được xây dựng từ các toán tử, các biểu thức đơn giản và các dấu ngoặc để biểu thị một giá trị hay một địa chỉ.

Chú ý: Có lệnh không cần toán hạng, có lệnh cần 1 hay 2 toán hạng. Nếu có 2 toán hạng thì giữa 2 toán hạng phải có dấu phảy (,). Toán hạng đầu gọi là toán hạng đích, toán hạng sau gọi là toán hạng nguồn.

Vùng chú thích (Comment) luôn bắt đầu bằng dấu chấm phẩy (;), dùng để ghi chú hay giải thích ý nghĩa của một hay nhiều câu lệnh để người lập trình dễ dàng tham khảo, đọc và hiểu chương trình. Assembler sẽ bỏ qua phần này khi dịch chương trình sang mã máy.

3.2.4. Các dạng hằng dùng trong assembly

Chương trình biên dịch Assembler cho phép dùng các dạng hằng số sau:

Nhị phân: gồm một dãy các chữ số 1 và 0 kết thúc bằng ký tự B, ví dụ: 101101001B.

Thập phân gồm một dãy các chữ số từ 0 đến 9, có hoặc không có ký tự D theo sau, ví dụ 345D hay 345.

Hexa gồm một dãy các chữ số từ 0 đến 9 và các ký tự từ A đến F (a đến f), theo sau bởi ký tự H. Ký tự đầu tiên phải là một chữ số.

Ví dụ : 7FFFH, 0E45H

Chú ý: Số 0 bắt buộc phải có để báo cho assembler biết E45H là một số, mà không phải là tên.

+ Chuỗi ký tự (xâu ký tự) : Gồm một dãy các ký tự bất kỳ nằm giữa 2 dấu (' hay giữa 2 dấu ("). Ví dụ : 'This is a string not a number'. Dấu (') hay (") có thể dùng như ký tự nếu nó xuất hiện 2 lần bên trong chuỗi. Ví dụ : 'I can"t go tonight' tương đương với "I can't go tonight".

3.2.5. Các chỉ dẫn trong assembly (Directive)

Giống như các lệnh assembly, directive cũng gồm 4 phần :

[Tên] Tên Directive [Toán hạng] [; Chú thích]

Mỗi phần được tách nhau bằng một hoặc nhiều ký tự ngăn cách.

Toán hạng có thể là tên tượng trưng, hằng, biến, biểu thức, các tên dành riêng tuỳ thuộc vào loại directive. Toán hạng dùng trong directive phải là một giá trị xác định khi dich.

Directive thường được dùng để định nghĩa tên tượng trưng, khai báo dữ liệu và biến, khai báo mảng và chương trình con, đánh dấu cuối mảng, cuối chương trình con hay kết thúc chương trình...

Macro assembler có rất nhiều directive khác nhau, nói chung ta có thể chia làm các nhóm chính sau:

a. Nhóm định nghĩa tên tượng trưng (Symbol definition directive) gồm 2 directive ***EQU*** và **=** . Cú pháp:

Name EQU <text> hoặc Name EQU expression.

Chức năng: EQU định nghĩa tên tượng trưng name và gán văn bản text hay trị của biểu thức expression 16 bit cho name. Ví dụ:

```
Table     EQU  [ BX] [SI]   ; một toán hạng
F10       EQU  68           ; một hằng số
Count     EQU  CX           ; một toán hạng
Sector    EQU  512          ; một hằng số
move      EQU  MOV          ; một từ khoá
 x        EQU  count        ; một tên tượng trưng khác
RT  EQU  <run-time>  ; một chuỗi
```

Directive = có cú pháp:

Name = expression

Chức năng: Directive = cho phép định nghĩa hay ***định nghĩa lại*** tên tượng trưng name và gán trị của biểu thức exptession 16 bit cho name. Directive = giống như EQU nhưng nó có thể gán lại trị cho name nhiều lần. Ví dụ:

```
dong = 10
MOV   AL, dong
dong  = 20
```

b. Nhóm khai báo dữ liệu (Data definition directive):

Nhóm directive này cho phép khai báo vùng nhớ dành cho các dữ liệu cần sử dụng trong chương trình. Các dữ liệu có thể là số, chuỗi hay một biểu thức có trị xác định.

Khai báo biến (byte, word, doubleword)

Biến là một vùng nhớ có tên dùng để chứa dữ liệu. Nếu chương trình muốn truy xuất vùng nhớ này ta chỉ cần dùng tên biến trong câu lệnh. Cú pháp:

[name] ***DB*** data [,....]

[name] ***DW*** data [,....]

[name] ***DD*** data [,....]

DB, DW, DD dùng để khai báo dữ liệu dạng byte, dạng word (2 byte) và dạng doubleword (4 byte).

Name là một tên dùng để truy xuất vùng dữ liệu.

data: qui định giá trị ban đầu của vùng dữ liệu. Ví dụ

```
B_max     DB    255
BS_min    DB    -128
W_max     DW    65535
WS_min    DW    -32767
DD_max    DD    4294967294
DDS_min   DD    -2147483647
MSG  DB   'Assembly language'
DB   'Hello$'
```

Khai báo một mảng dữ liệu trong bộ nhớ:

```
B_table DB   1,2,3,4,5
             DB    6, 7,8,9,10
W_table      DW    100, 200, 0, - 300,1024
```

Toán tử DUP: (duplicate): Dùng để lặp lại các dữ liệu với số lần qui định bởi count. Cú pháp: ***count DUP (data[,...])***. Chú ý: Toán tử DUP có thể lồng nhau.Ví dụ

Tạo một mãng dữ liệu gồm 500 bytes có trị 0.

B_ mem500 DB 500 dup (0)

Tạo một mãng dữ liệu gồm 500 word có trị 0FFFFH

W_mem500 DW 500 dup (0FFFFH).

Toán tử ?:

Muốn khai báo một biến hay một mảng mà không cần khởi đầu giá trị của nó ta dùng toán tử?, lúc đó trị của các biến hay các mảng này là không xác định, nó phụ thuộc vào nội dung của biến hay mảng đó trong bộ nhớ lúc assembler dịch.Ví dụ:

mem8 DB ? ; khai báo 1 byte trống trong bộ nhớ

mem16 DW ? ; khai báo 2 byte trống trong bộ nhớ

Bmem500 DB 500 DUP (?); Khai báo vùng nhớ gồm 500 word

Khai báo biến con trỏ (pointer). Biến con trỏ là biến dùng để chứa địa chỉ ô nhớ. Biến con trỏ có thể là near hay far. Biến con trỏ dạng near chỉ chứa phần địa chỉ offset, khai báo như một biến dạng word bằng directive DW. Biến con trỏ dạng far chứa cả địa chỉ mảng và địa chỉ offset, khai báo như một biến dạng doubleword bằng directive DD. Giả sử ta có nhãn NEXT:

NearNext DW offsetnext; gán địa chỉ offset của nhãn next vào biến NearNext.

FarNext DW next; gán địa chỉ offset và địa chỉ mảng của nhãn next tương ứng vào phần thấp và phần cao của biến con trỏ 32 bit FarNext.

Khai báo biến nhãn (labei variable).

Directive LABEL có thể dùng để định nghĩa một tên biến. LABEL được dùng khi ta muốn truy xuất đến cùng một vùng nhớ nhưng dùng các dạng dữ liệu khác nhau.Ví dụ khai báo một mảng mà có thể truy xuất bằng cả 3 dạng dữ liệu byte, word, doubleword.

warrray LABEL word

darrray LABEL dword

barray DB 100 dup (?)

Vậy với cùng một vùng nhớ, có 3 tên là waray (nếu xem vùng này gồm 50 dword hoặc 25 dword nếu xem vùng này là darray) và barray (nếu xem vùng này gồm 100 byte).

c. Nhóm khai báo mảng và chương trình con

Directive SEGMENT VÀ ENDS dùng để khởi đầu một mảng chứa một dạng cơ sở dữ liệu (có thể là mảng chứa chương trình, có thể là mảng chứa dữ liệu, có thể là mảng chứa ngăn xếp ...). Cú pháp của nó có dạng :

name SEGMENT [align_ type] [combine_type] ['class']

....

name ENDS

Như vậy Directive ***segment*** và ***ends*** dùng để chia chương trình nguồn thành các mảng phân biệt khi cần thiết. có 4 kiểu mảng là CODE, DATA, EXTRA, STACK.

name định nghĩa tên của mảng. Các tên mảng có thể trùng tên nhau.

Các tham số align - type, combine - type, class chỉ có ảnh hưởng khi chương trình assembler khai báo nhiều hơn một mảng.

Align - type: Xác định vị trí bắt đầu của mảng trong bộ nhớ. Nếu Align-type được khai báo là :

byte chỉ rằng mảng có thể bắt đầu tại một vị trí bất kỳ trong bộ nhớ.

word chỉ rằng mảng phải bắt đầu tại địa chỉ chẵn (địa chỉ chia hết 2)

para chỉ rằng mảng phải bắt đầu tại địa chỉ para (địa chỉ chia hết 16).

Page chỉ rằng mảng phải bắt đầu tại địa chỉ page (địa chỉ chia hết 256).

Nếu không có align-type thì trị ngầm định là PARA.

Combinre-type quy định cách liên kết các mảng có cùng tên. Nếu Combine-type được khai báo là :

Public chỉ rằng phải ghép tất cả các mảng có cùng tên thành một mảng chung duy nhất. Tất cả các nhãn và các biến trong mảng chung đều dùng chung một địa chỉ mảng, địa chỉ offset được tính từ điểm bắt đầu của mảng chung.

Stack chỉ rằng phải ghép tất cả các mảng có cùng tên thành một mảng chung duy nhất. Mảng chung này được chỉ bởi thanh ghi mảng SS, còn thanh ghi SP được xác định bởi chiều dài của mảng chung. Mảng stack phải được khai báo theo kiểu STACK.

Common chỉ rằng phải ghép tất cả các mảng có cùng tên chồng lên nhau tại cùng một địa chỉ. Độ dài của mảng chung này chính bằng độ dài lớn nhất của tất cả các mảng.

At address - các nhãn hay biến khai báo trong mảng đều có địa chỉ mảng do address qui định. Mảng dạng At khi asembler dịch không tạo thêm mã cho

chương trình mà chỉ dùng để tham chiếu đến các ô nhớ chứa mã hay dữ liệu đã tồn tại trong bộ nhớ. Mảng dạng At có thể dùng trong tập tin dạng COM.

Nếu không có combine-type thì các mảng cùng tên không liên kết nhau. Mỗi mảng sẽ có một địa chỉ mảng riêng khi được nạp vào bộ nhớ.

Class xác định thứ tự của các mảng khi được nạp vào bộ nhớ. Các mảng khác tên lớp được nạp vào bộ nhớ theo thứ tự tuần tự hay theo thứ tự tăng dần của tên lớp tùy theo cách liên kết (LINK).

Ví dụ để định nghĩa một đoạn dữ liệu:

```
Data_seg    SEGMENT
            a  DB  0
            b DB   0
            c  DW ?
Data_seg    ENDS
```

Định nghĩa một đoạn chương trình:

```
Code_seg    SEGMENT
...
            MOV   CL , 2
            SHL   AX, CL
            MOV   BX , AX
            ...
Code_seg ENDS
```

Directive ***PROC và ENDP.*** Cặp chỉ dẫn này dùng để khởi đầu một modul chương trình dưới dạng thủ thục. Cú pháp của nó có dạng:

```
Name PROC [type]
...
Name  ENDP
```

Directive PROC và ENDP dùng để khai báo một chương trình con, nó đánh dấu điểm bắt đầu và điểm cuối của một chương trình con.

Nếu chương trình con ở dạng near: chỉ có thể được gọi trong cùng mảng chứa chương trình đó. Ví dụ:

```
Code_seg    SEGMENT
ASSUME  CS : Code_seg
        ...
CALL proc_A; gọi chương trình con proc_A trong cùng mảng; có tên
Code_seg
        ...
proc_A PROC near  ; khai báo chương trình con có tên proc_A
        ...
   RET
proc_A ENDP
Code_seg    ENDS
```

Nếu chương trình con ở dạng far: có thể được gọi trong mảng khác với mảng chứa chương trình con đó. Ví dụ:

```
Code_seg      SEGMENT
ASSUME CS : Code_seg
        ...
CALL far proc_A       ; gọi hương trình con proc_A trong mảng
                      ; Code_seg 1
        ...
Code_seg      ENDS
Code_seg 1    SEGMENT
ASSUME   CS: Code_seg 1
        ...
proc_A  PROC  far
        .....
RET
pro_A endp
Code_seg 1 ENDS
```

d. Nhóm tham chiếu bên ngoài (external reference directive) với Directive PUBLIC và EXTRN.

Đối với những chương trình lớn, thường phải chia chương trình ra thành nhiều phần (module) nhỏ, mỗi phần có thể nằm tách rời trên những file nguồn khác nhau. Sau khi asembler dịch từng phần sang file mã máy dạng Object tương ứng, chúng ta mới LINK các tập tin Object đó lại thành một chương trình duy nhất mà hệ vi xử lý có thể thực hiện được.

Do đó muốn một tên biến, nhãn hay tên tượng trưng trong phần này có thể dùng được trong các phần khác ta phải khai báo chúng bằng directive PUBLIC:

PUBLIC name [,name] ...

name là tên biến, nhãn hay tên tượng trưng.

Khi một file muốn sử dụng các biến, nhãn hay tên tượng trưng đã được khai báo bằng directive PUBLIC trong các file khác, ta phải dùng directive EXTRN để báo cho asembler biết là chúng đã được khai báo trong các file khác.

EXTRN name: type [,name:type]...

name là tên biến, nhãn hay tên tượng trưng được khai báo bằng directive PUBLIC trong các phần khác:

Nếu name là một biến thì type có thể là byte, word hay dword.

Nếu name là một nhãn hay một chương trình con thì type có thể là near hay far

Nếu name là một hằng số được định nghĩa bằng directive EQU hay = thì type phải là ABS.

Ví dụ: giả sử có 2 file. File 1 có nội dung sau:

```
Code_Seg      SEGMENT
ASSUME CS: Code_Seg
EXTRN  bsort:FAR
      ....
CALL FAR PTR  bsort
      ....
Code_Seg ENDS
END
```

File 2 có nội dung như sau:

```
PUBLIC  bsort
Code_Seg SEGMENT
ASSUME  CS : Code_Seg
      ......
bsort  PROC  FAR
      ....
RET
bsort ENDP
      ...
Code_Seg ENDS
END
```

Directive ***include*** có cú pháp:

INCLUDE filename

Filename là tên file nguồn chứa các phát biểu asembler. Directive INCLUDE đọc toàn bộ nội dung file có tên qui định bởi filename và file nguồn chứa directive include đó khi dịch.Ví dụ:

INCLUDE fileio.mac

e. Nhóm directive điều khiển (control directive):

Directive END có cú pháp

END [start_address]

Directive END báo cho asembler biết chương trình kết thúc tại vị trí END này. Start_address là một nhãn xác định điểm bắt đầu của chương trình. Chú ý rằng mỗi file nguồn phải chứa một directive END.

Nếu chương trình gồm nhiều phần nằm tách rời trên những file khác nhau, thì ta chỉ có thể định nghĩa start_address sau directive END trong phần chính (main module).

Directive EVEN làm cho chương trình chạy nhanh hơn. Ta biết CPU 80286 dùng BUS dữ liệu 16 bit, nên có thể chuyển 16 bit dữ liệu cùng một lúc. Tuy nhiên việc chuyển 16 bít dữ liệu bắt đầu tại địa chỉ lẻ sẽ thực hiện chậm hơn tại địa chỉ chẵn. Vì vậy muốn chương trình thực hiện nhanh hơn, các dữ liệu cần truy xuất nên lưu tại địa chỉ chẵn.

Thường trong chương trình ta nên khai báo các dữ liệu dạng doubleword trước tiên, sau đó đến các dữ liệu dạng word rồi cuối cùng mới đến các dữ liệu dạng byte. Directive EVEN dùng để báo cho asembler biết các dữ liệu khai báo ngay sau directive EVEN sẽ được nạp vào bộ nhớ tại địa chỉ chân.

f. Mode directive: Vì Macro Assembler là chương trình dịch cho cả CPU 8086/ 8088/ 80286/ 80386..., do đó mode directive dùng để báo cho assembler biết chương trình muốn dịch đối với CPU loại nào, nếu không dùng mode directive thì MASM sẽ dịch chương trình đối với CPU 8086/8088.

.8086 chỉ cho phép dịch các lệnh của CPU 8086/8088 và bộ đồng xử lý toán học 8087.

.286 Cho phép dịch các lệnh của CPU 8086/8088/80286 và bộ đồng xử lý toán học 80287.

.386 Cho phép dịch các lệnh của CPU 8086/88/286/386 và bộ đồng xử lýtoán học 80387.

Chú ý: Mode directive phải được dùng bên ngoài mảng. Ví dụ muốn dịch các lệnh mới của CPU 80286 ta phải dùng directive:

.286

ở đầu chương trình.

h. Comment directive: Comment directive dùng để ghi chú, giải thích ý nghĩa của một đoạn chương trình. assembler khi dịch sẽ bỏ qua phần này. Cú pháp có dạng

COMMENT * [text]

text

*[text]: Tất cả các văn bản text giữa 2 dấu (*) và văn bản trên dòng chứa dấu * thứ hai sẽ là phần ghi chú. assembler sẽ bỏ qua phần này khi dịch.

3.2.6. Các toán tử (operator) dùng trong assembler

Sự khác nhau giữa toán tử và lệnh là:

Toán tử điều khiển việc tính toán các giá trị hằng xác định lúc dịch chương trình.

Lệnh điều khiển sự tính toán các giá trị không xác định được lúc dịch chương trình. Chỉ đến khi chương trình thực hiện thì các giá trị mới được xác định.

Ví dụ: Toán tử + điều khiển phép cộng khi dịch và lệnh cộng ADD điều khiển phép cộng khi chương trình thực hiện.

Assembler cung cấp các toán tử sau:

Các toán tử số học:

Toán tử	*Cú pháp*	*Công dụng*
+	+exp	Dấu dương
-	-exp	Dấu âm
*	exp1*exp2	Nhân
/	exp1/exp2	Chia
mod	exp1 mod exp2	Phần dư
+	exp1+ exp2	Cộng
-	exp1- exp2	Trừ
shl	exp shl count	Dịch exp sang trái *count* bit
shr	exp shr count	Dịch exp sang phải *count* bit.

Trong đó exp, exp1, exp2 là các biểu thức hằng, count là số nguyên.

Ví dụ:

```
MOV     BX , (80 * 4 +10)*2
MOV  AX , 01110100B shl 2
MOV     dl , 300 mod 8
```

Các toán tử logic:

Toán tử	*Cú pháp*
not	not exp
and	exp1 and exp2
or	exp1 or exp2
xor	exp1 xor exp2

Ví dụ:

```
MOV  AL , 8 or 4 and 2 ;
MOV  AL , not (20 xor 0011100B).
```

Nhóm các toán tử quan hệ

Các toán tử quan hệ so sánh 2 biểu thức và cho trị true (-1) nếu điều kiện của toán tử thoả mãn, ngược lại cho trị false (0).

Toán tử	*Cú pháp*	*cho trị*
EQ	exp1 EQ exp2	true nếu exp1= exp2
NE	exp1 NE exp2	true nếu exp1<> exp2
LT	exp1 LT exp2	true nếu exp1< exp2
LE	exp1 LE exp2	true nếu exp1<= exp2
GT	exp1 GT exp2	true nếu exp1> exp2
GE	exp1 GE exp2	true nếu exp1>= exp2

Ví dụ:

```
MOV  AX , 4 EQ 3     ; cho trị 0
MOV  AX , 4 NE 3     ; cho trị -1
MOV  AX , 4 LT 3     ; cho trị 0
```

Nhóm cung cấp thông tin về biến và nhãn

+ Toán tử SEG:

```
SEG  exprression
```

Cho địa chỉ mảng của biểu thức expression. expression có thể là một biến, nhãn, tên SEGMENT, tên GROUP hay các toán hạng bộ nhớ khác.

Ví dụ: Nạp địa chỉ segment và offset của biến table vào cặp thanh ghi DS:DX

```
table   DB          ?
MOV     AX, SEG     table
MOV     DS, AX
MOV     DX, OFFSET table
```

+ Toán tử (:) (segment overide operator) có cú pháp

segment: expression

Segment có thể là tên các thanh ghi mảng CS, DS, ES, SS hay tên mảng, tên GROUP đã khai báo expression có thể là một hằng số hay một biểu thức.

Chức năng: toán tử (:) quy định cách tính địa chỉ của một biến hay một nhãn đối với mảng segment được chỉ ra. Chú ý rằng nếu toán tử (:) dùng chung với các toán tử chỉ số [] thì segment phải đặt ngoài toán tử [].

Ví dụ: Toán hạng [ES : DI] viết sai, phải viết lại là ES :[DI].Ví dụ :

```
MOV   AX  , ES : [BX +4]
MOV   AX  , Data_seg: count
```

+ Toán tử $: cho địa chỉ offset của phát biểu chứa toán tử $ đó, thường toán tử $ dùng để tính độ dài của một chuỗi.

Ví dụ:

```
str      DB 'To count every byte in a string'
         DB 'especially if you might change it later'
lenstr EQU    $ - offset str
```

+ toán tử type:

TYPE expression

cho trị biểu thị dạng của biểu thức expression

Nếu expression là biến thì trị 1 biểu thị dạng byte, 2-word, 4-dword.

Nếu expression là nhãn thì trị OFFFFh biểu thị dạng near và OFFFEH biểu thị dạng far.

Nếu expression là hằng thì TYPE cho trị O.

Ví dụ:

```
var       DW  ?
array     DD   10 DUP (?)
str       DB 'this is  string'
MOV       AX, TYPE var        ; cho trị 2
MOV       AX, TYPE array      ; cho trị 4
MOV       AX, TYPE  str       ; cho trị 1
```

+ Toán tử Length:

```
LENGTH          var
```

Cho số các đơn vị mà biến var xin cấp phát. Ví dụ:

```
table  DW  500 DUP (?)
MOV  CX, LENGTH  table  ; cho trị 500
```

Nhóm thuộc tính

+ Toán tử PTR (pointer):

```
type  PTR expression
```

cho phép thay đổi dạng của biểu thức expression

- Nếu expression là một biến hay một toán hạng bộ nhớ thì type có thể là byte, word hay dword.

- Nếu expression là một nhãn thì type có thể là near hay far.

Ví dụ:

```
stuff   DD  ?
table   DW  500 dup  (?)
MOV  AL, BYTE  PTR stuff; nạp byte đầu của mảng table vào AL
MOV  BX, WORD  PTR stuff; nạp nội dung 2 byte thấp của biến
                          Stuff vào thanh ghi BX
MOV  BYTE PTR [BX], 0; nạp trị 00 vào ô nhớ có địa chỉ xác định
                       bởi thanh ghi BX
MOV  WORD PTR [BX], 0 ; nạp trị 0000 vào vùng nhớ 2 byte có
                        địa chỉ xác định bởi thanh ghi BX.
```

+ Toán tử high và low:

```
HIGH        expression; Cho trị của byte cao và byte thấp của
LOW         expression; biểu thức expression.
```

Expression phải là một trị hằng xác định khi dịch. Ví dụ:

```
Const  0ABCDH
MOV   AL,  LOW   const  ; trị CD →AL
MOV  AH ,  HIGH const  ; trị AB→AH
```

3.3. CHƯƠNG TRÌNH BIÊN DỊCH MACRO ASSEMBLER 5.1

Trong tài liệu này chúng ta sử dụng chương trình dịch Macro assembler của hãng Microsoft version 5.1, nó gồm 3 file chính:

MASM.EXE: Dịch file nguồn sang tập tin mã máy dạng Object, tương ứng (có tên mở rộng OBJ).

LINK.EXE: Liên kết các file mã máy dạng Object thành file chạy dạng EXE.

EXE2BIN.EXE : Chuyển đổi file có tên mở rộng EXE viết theo dạng COM thành file chạy dạng COM.

MASM có thể dịch một chương trình nguồn sang file mã máy dạng OBJ bằng 2 cách:

Cách dùng tham số trên dòng lệnh tại dấu nhắc DOS.

Cách đối thoại trực tiếp từ bàn phím.

Cách dịch dùng tham số:

MASM [option] sourcefile [, [objfile] [,[listingfile]

option là tham số tuỳ chọn của MASM có thể tra cách dùng option trong tài liệu "Microsoft Macro asembler".

Sourcefile là tên file nguồn cần dịch. Nếu không ghi rõ tên mở rộng thì MASM se tự động lấy tên mở rộng là .ASM.

Objfile là tên file object cần tạo. Nếu không có tên object thì MASM dùng tên file nguồn và tên mở rộng là .obj.

Listingfile là tên file listing. Nếu không có tên mở rống thì MASM sẽ tự động lấy tên mở rộng là .LST.

Crossreffile là tên file cross reference .Nếu không có tên mở rộng thì MASM sẽ tự động lấy tên mở rộng là .CRF.

Dấu;

- Nếu dùng dấu; sau sourcefile thì MASM sẽ chọn tên objfile định sẵn, không tạo tập tin listing và cross reference.
- Nếu dùng dấu ; sau objfile thì MASM sẽ không tạo tập tin listing và cross reference.
- Nếu dùng dấu; sau listingfile thì MASM không tạo tập tin cross reference.

Ví dụ:

MASM file.ASM, file.OBJ, file.LST, file.CRF; tương đương với

MASM file, , ;

Dịch file nguồn có tên file.asm thành file mã máy file.obj, đồng thời MASM cũng tạo file listing file.lst và file cross refrence file.crf

MASM file ;

Dịch file nguồn thành tập tin file.obj nhưng không tạo file listing lẫn file cross reference.

Cách dịch bằng đối thoại trực tiếp từ bàn phím:

MASM

MASM sẽ xuất các thông báo sau ra màn hình và chờ ta đối thoại từng dòng một.

Source filename [ASM]:

Giải thích:

- Dòng đầu chờ ta gõ tên file hợp ngữ nguồn cần dịch, nếu không ghi tên mở rộng thì MASM sẽ lấy tên mở rộng là ASM.
- Dòng thứ hai chờ ta gõ tên file dạng OBJ. Nếu không ghi thì MASM sẽ lấy tên file nguồn và tên mở rộng là OBJ.
- Dòng ba chờ ta gõ tên file listing. Nếu không thì thì MASM sẽ không tạo file listing.
- Dòng cuối chờ ta gõ tên file cross reference. Nếu không ghi thì MASM sẽ không tạo file cross refrerence.

***Chương trình liên kết LINK*:**

Cú pháp của chương trình LINK:

LINK [option] objectfile1 [objecttile2] [...] [,executablefile]

LINK dùng để liên kết các file object có tên objectfilel, objectfilel2 ... thành file chạy dạng EXE có tên do executablefile qui định. Nếu không có tham số executablefile thì LINK sẽ lấy tên của file objectfile1.

Option là tham số tuỳ chọn của LINK có thể tra cách dùng option trong tài liệu "Microsoft Macro asembler"

Chương trình exe2bin.exe

Cú pháp : ***exe2bin.exe exefile [binaryfile]***

Chuyển đổi file có tên do exefile qui định thành file dạng COM có tên do binaryfile qui định.

Chú ý:

file exefile phải có tên mở rộng EXE.

Nếu bianryfile không có phần tên mở rộng thì MASM sẽ tự động lấy tên mở rộng là .BIN.

Ví dụ: EXE2BIN file file.com sẽ chuyển file file.exe thành file dạng com file.com.

3.4. TẬP LỆNH CỦA BỘ VI XỬ LÝ 80X86

Tập lệnh của bộ vi xử lý 80x86 rất phong phú, cho phép xây dựng các chương trình có chức năng từ đơn giản đến phức tạp, từ quy mô nhỏ đến quy mô lớn. Tập lệnh của bộ vi xử lý 80x86 có thể chia thành nhiều nhóm. Chúng ta sẽ lần lượt xét chức năng của từng nhóm lệnh này.

Trước hết chúng ta phải thống nhất các qui ước khi viết cú pháp một lệnh assembly. Các qui ước này là:

[ ] : Những tham số đặt trong 2 dấu [ ] là tuỳ chọn (có thể có hoặc không).

Reg : Chỉ toán hạng thanh ghi (8 bit hay 16 bit).

Reg8 : Chỉ toán hạng thanh ghi 8 bit.

Reg16 : Chỉ toán hạng thanh ghi 16 bit.

Mem : Chỉ toán hạng bộ nhớ 8 bít hay 16 bit

Mem8 : Chỉ toán hạng bộ nhớ 8 bit

Mem16 : Chỉ toán hạng bộ nhớ 16 bit

Mem32 : Chỉ toán hạng bộ nhớ 32 bit

Immed : Chỉ toán hạng trực tiếp (8 bit hay 16 bit).

Immed8: Chỉ toán hạng trực tiếp 8 bit .

Immed16: Chỉ toán hạng trực tiếp 16 bit.

SegReg : Chỉ toán hạng thanh ghi mảng.

3.4.1. Nhóm lệnh chuyển dữ liệu

Lệnh MOV (move):

Toán hạng nguồn Source có thể là Reg, Mem hay Immed.

toán hạng đích Dest chỉ có thể là Reg hay Mem.

Chức năng : Chuyển nội dung toán hạng source vào tóan hạng dest. Kích thước dữ liệu có thể là 8 bit hay 16 bit.

Trường hợp 1 : Chuyển dữ liệu giữa hai thanh ghi:

Reg8 <-- reg8 . Ví dụ:

MOV AL, AH; chuyển nội dung thanh ghi 8 bit AH vào AL

Reg16 <-- reg16. Ví dụ:

MOV BX, SI; chuyển nội dung thanh ghi 16 bit SI vào BX.

Trường hợp 2 : Chuyển dữ liệu giữa thanh ghi và bộ nhớ.

Mem8 <-- reg8. Ví dụ:

MOV [BX], AL; chuyển AL vào ngăn nhớ có địa chỉ là nội dung của thanh ghi BX

MOV DS : [150h], AL

MOV ES : [SI], DL

Reg8 <-- mem8. Ví dụ:

MOV AL, mem [BX]

MOV AH, ds: [10H]

men16 <-- reg16.Ví dụ:

MOV [BX], AX

MOV varw,AX; varw là một biến dạng word.

MOV CS : [SI+BX], DX

Trường hợp 3: Gán giá trị hằng vào thanh ghi hay bộ nhớ.

reg8 <-- immed8. Ví dụ:

MOV AL, 0B5h

Mem8 <-- inned8. Ví dụ:

MOV mem[BX],-1

MOV byte ptr ds: [150h],10h

Reg16 <-- immed16.Ví dụ:

MOV AX,0B800H

mem16 <-- immed16.Ví dụ:

MOV count,2000

MOV word ptr [BX],5AB7H

Trường hợp 4: Chuyển dữ liệu giữa thanh ghi mảng và thanh ghi hay bộ nhớ.

segreg <-- reg16. Ví dụ:

MOV DS, AX

Segreg <-- mem16. Ví dụ:

MOV ES , screen

reg16 <-- segreg. Ví dụ:

MOV AX , CS

mem16 <-- segreg.Ví dụ :

MOV [BX + DI] , ES

Chú ý: Lệnh MOV không ảnh hưởng thanh ghi cờ. MOV không thể chuyển dữ liệu trực tiếp giữa hai toán hạng bộ nhớ với nhau, muốn chuyển ta phải dùng một thanh ghi trung gian. Ví dụ: Muốn chuyển dữ liệu 16 bit từ mem1 vào mem2, ta phải

```
MOV   AX, mem1
MOV   mem2, AX
```

MOV không thể chuyển trực tiếp một hằng vào một thanh ghi mảng, muốn chuyển ta phải dùng một thanh ghi trung gian. Ví dụ: Chuyển giá trị B800h vào thanh ghi DS, ta phải dùng một thanh ghi trung gian 16 bit

```
MOV   AX, 0B800H
MOV   DS  , AX
```

MOV không thể chuyển trực tiếp theo giữa hai thanh ghi mảng. Nó cũng không thể dùng thanh ghi mảng CS làm toán hạng đích trong lệnh MOV.

Lệnh XCHG (exchange):

XCHG dest, Source

Toán hạng source và dest chỉ có thể là Reg hay Mem.

Chức năng: Hoán chuyển nội dung của 2 toán hạng nguồn và đích. Ta chỉ có thể hoán chuyển giữa thanh ghi và thanh ghi hoặc giữa thanh ghi và bộ nhớ. Ví dụ:

```
XCHG  AX , BX       ; Hoán chuyển giữa 2 thanh ghi 16 bit
XCHG  AL , CH       ; Hoán chuyển giữa 2 thanh ghi 8 bit
XCHG  AX , [BX]     ; Hoán chuyển giữa  thanh ghi và bộ nhớ
XCHG  BX , mem      ; Hoán chuyển giữa  thanh ghi và bộ nhớ
XCHG mem , AX
```

Chú ý: Lệnh này không ảnh hưởng thanh ghi cờ. Không thể dùng lệnh này với các thanh ghi mảng.

Lệnh PUSH:

```
PUSH     source (reg16 hay mem16)
PUSH     immed16 (chỉ dùng với CPU 80286/386/486...)
```

Chức năng: Dùng để cất một hằng hay nội dung một thanh ghi 16 bit hay nội dung một toán hạng bộ nhớ 16 bit vào STACK.

Lệnh PUSH giảm thanh ghi SP đi 2 đơn vị và chuyển nội dung 16 bit của toán hạng nguồn vào đỉnh của STACK. Đỉnh STACK được xác định bởi cặp thanh ghi SS:SP. Ví dụ:

```
PUSH  SI
PUSH  DS
PUSH  CS
PUSH  SP
PUSH  mem [BX][SI]
PUSH  [BX + SI]
PUSH  147EH (chỉ dùng với CPU 80286/80386/80486/80586 ...)
```

Lệnh POP:

POP dest (reg16 hay mem16)

Chức năng : POP dùng để lấy dữ liệu 16 bit trong STACK (được xác định bởi cặp thanh ghi SS:SP) vào toán hạng dest.

Lệnh POP tăng thanh ghi SP 2 đơn vị để chỉ đến đỉnh mới của STACK. Ví dụ:

```
POP   SI
POP   DS
POP   CS
POP   SP
POP   mem[BX][SI]
POP   [BX + SI]
```

Dùng lệnh PUSH và POP để chuyển thanh ghi mảng CS vào thanh ghi mảng DS:

```
PUSH  CS
POP   DS
```

Hoạt động của STACK: hình 3.1a,b,c giải thích cách hoạt động của STACK trước và sau một lệnh PUSH, và sau một lệnh POP.

Chú ý: Có thể lưu nhiều dữ liệu vào STACK bằng cách dùng một dãy lệnh PUSH, vì PUSH cất dữ liệu trên đỉnh của STACK nên ta phải POP các dữ liệu ra khỏi STACK theo thứ tự ngược lại khi PUSH (nguyên tắc LIFO- vào sau cùng ra trước tiên).

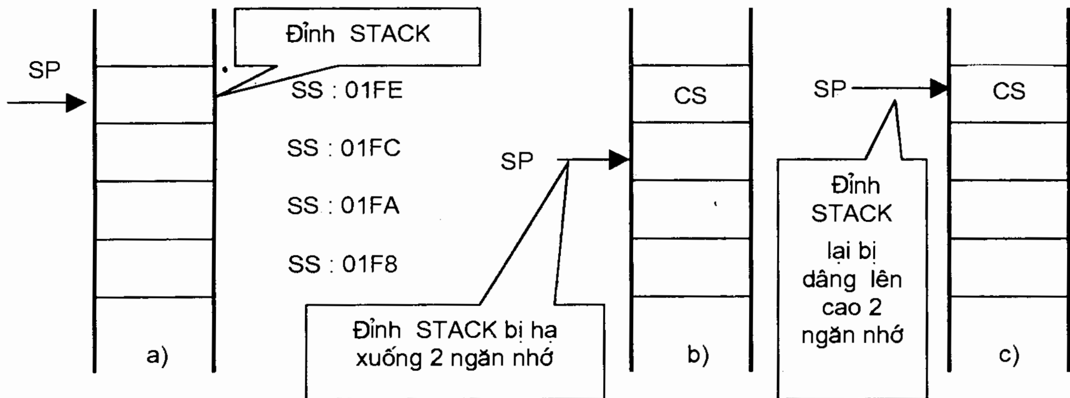
Ví dụ:

Lưu nội dung 4 thanh ghi AX, BX, CX, DX vào STACK

```
PUSH AX
PUSH BX
PUSH CX
PUSH DX
```

Khi muốn lấy lại nội dung các thanh ghi AX, BX, CS,DX đã lưu trong STACK ta phải POP chúng ra theo thứ tự ngược lại

```
POP   DX
POP   CX
POP   BX
POP   AX
```



Hình 3.1. a) STACK trước khi PUSH CS; b) STACK sau khi PUSH CS; c) STACK sau khi POP DS.

Lệnh PUSHA (PUSH ALL) chỉ dùng với CPU 80286/386/486...

Chức năng: PUSH lưu tất cả các thanh ghi theo thứ tự AX, CX, DX, BX, SP, BP, SI, DI lần lượt vào STACK.

Chú ý: Lệnh PUSHA không cất nội dung các thanh ghi mảng và thanh ghi cờ vào STACK.

Lệnh POPA (POP ALL) chỉ dùng với CPU 80286/386...

Chức năng: POPA khôi phục nội dung tất cả các thanh ghi theo thứ tự ngược lại DI, SI, BP, SP, SP, BX, DX, CX, AX lần lượt từ STACK.

Lệnh XLAT (translate).

XLAT (source-table)

Chức năng: Chuyển nội dung của ô nhớ nằm trong một bảng các ô nhớ 8 bit vào thanh ghi AL. Trong đó bảng có địa chỉ bắt đầu xác định bởi cặp thanh ghi DS:BX, Vị trí offset của byte nhớ trong bảng được xác định bởi thanh ghi AL

XLAT Tương đương với các lệnh sau:

```
MOV     AH, 0
MOV     SI,AX
MOV     AL,[BX+SI]
```

Bảng các ô nhớ chỉ dài tối đa 256 bytes

Thông thường DS chứa địa chỉ mảng của bảng, nhưng ta có thể dùng thanh ghi mảng khác bằng cách ghi tên thanh ghi mảng cần dùng trong source-table.Ví dụ, muốn dùng thanh ghi mảng CS để xác định địa chỉ mảng của bảng table thì ta dùng lệnh ***XLAT CS: table***.

Ví dụ: Cho hai vùng nhớ MEM1 và MEM2 nằm trong mảng chỉ bởi DS. Mỗi vùng có chiều dài 3 byte, vùng nhớ MEM1 gồm các số có trị trong khoảng từ 0 đến 15. Viết đoạn chương trình chuyển đổi nội dung các trị trong vùng nhớ MEM1 thành những ký tự từ '0' đến '9' và từ 'A' đến 'F' tương ứng trong vùng nhớ MEM2.

Giải:

```
MOV        SI ,    offset MEM1
MOV        DI ,    offset MEM2
MOV        BX ,    offset bangkytu
MOV        AL ,    [SI]
XLAT
MOV        [DI], AL
MOV        AL , [SI+1]
XLAT
MOV        [DI+1], AL
MOV        AL ,  [SI+2]
XLAT
MOV        [DI+2], AL
Bangkytư   db   '0123456789ABCDEF'
```

3.4.2.Nhóm lệnh chuyển địa chỉ

Lệnh LEA: (Load Effective address)

LEA reg16, mem16

Chức năng: Chuyển địa chỉ offset của một toán hạng bộ nhớ mem16 vào thanh ghi đích reg16. Ví dụ:

```
LEA    BX  , ds:[500H]       ; chuyển 500H vào BX.
LEA    SI  , table ; chuyển địa chỉ offset của table vào thanh ghi SI.
LEA    BX  ; table  [SI] ; chuyển địa chỉ offset của bảng table + nội
                         ; dung của thanh ghi SI vào thanh ghi BX.
```

Lệnh LDS: (Load Pointer using DS)

LDS reg16, mem32

Chức năng: Chuyển nội dung toán hạng bộ nhớ mem32 vào cặp thanh ghi 16 bit, phần cao 16 bit của mem32 được nạp vào thanh ghi DS, còn phần thấp được nạp vào thanh ghi reg16 trong lệnh.

Lệnh LDS BX , MEM [SI] tương đương với:

MOV BX , MEM [SI]

MOV AX , MEM [SI +2]

MOV DS , AX

Lệnh LES : (Load pointer using ES)

LES reg16, mem32

Chức năng: giống như lệnh LDS nhưng dùng thanh ghi mảng ES thay cho thanh ghi mảng DS.

3.4.3.Nhóm lệnh chuyển thanh ghi cờ

Lệnh LAHF: (Load AH from flag)

LAHF

Chức năng: Chuyển phần thấp của thanh ghi cờ gồm các cờ SF, ZF, AF, PF và CF tương ứng vào các bit 7,6,4,2 và 0 của thanh ghi AH, các bit còn lại 5,3 và 1 không thay đổi.

Lệnh SAHP: (Store AH into Flag)

SAHF

Chức năng: Chuyển các bit 7,6,4,2 và 0 của thanh ghi AH tương ứng vào các cờ SF, ZF,AF,PF,CF của thanh ghi cờ. Các cờ còn lại OF,DF,IF,TF không bị ảnh hưởng.

Lệnh PUSHF:

PUSHF

Chức năng: PUSHF giảm thanh ghi SP 2 đơn vị và chuyển nội dung của phần tử trên đỉnh STAK vào thanh ghi cờ.

3.4.4. Nhóm lệnh chuyển dữ liệu qua cổng

CPU giao tiếp và điều khiển các thiết bị ngoại vi thông qua các cổng vào ra (I/O port), mỗi cổng được xác định đơn trị bởi một địa chỉ cổng 16 bit.

CPU gửi dữ liệu hoặc thông tin điều khiển đến cổng bằng cách chỉ đến địa chỉ cổng đó, còn cổng nhận thông tin và nếu cần, cổng trả lời bằng cách chuyển dữ liệu hoặc thông tin trạng thái trở lại cho CPU. Tuỳ chức năng của từng cổng,

các cổng có thể: chỉ đọc được (Input port) hoặc chỉ ghi được (output port) hoặc vừa đọc vừa ghi được (Input/outout port)

Lệnh IN:

IN AL , port8

Hay IN AL , DX

Chức năng: Đọc một dữ liệu 8 bit từ một cổng nhập vào thanh ghi AL. Nếu địa chỉ của cổng có giá trị trong khoảng từ 0 đến FFh thì địa chỉ đó có thể viết trực tiếp trong câu lệnh, còn trong trường hợp địa chỉ của cổng có giá trị > FFh thì ta phải dùng thanh ghi DX để định địa chỉ cổng.

Ví dụ:

```
IN      AL  , 61H      ; đọc một byte từ cổng 61h
MOV     DX  , 03f8H
IN      AL  , DX       ; đọc một byte từ cổng 03f8H.
```

Lệnh OUT:

OUT port8 , AL

Hay OUT DX , AL

Chức năng : xuất dữ liệu 8 bit từ thanh ghi AL ra cổng xuất. Nếu địa chỉ của cổng có giá trị trong khoảng từ 0 đến FFh thì địa chỉ đó có thể viết trực tiếp trong câu lệnh, còn trong trường hợp địa chỉ của cổng có giá trị > FFh thì ta phải dùng thanh ghi DX để định địa chỉ cổng.

Ví dụ:

```
OUT   61H   , AL    ; gởi một byte ra cổng 61h
MOV   DX    , 03f8H
OUT   DX    , AL    ; gửi một byte ra cổng 03f8H.
```

3.4.5. Nhóm lệnh chuyển điều khiển

Lệnh nhảy không điều kiện JMP (jump) có cú pháp:

JMP Label hoặc

JMP Target (reg hoặc mem)

Chức năng: chuyển điều khiển (CS:IP) tới vị trí mới được xác định bởi Label hay nội dung của Target.

Dạng1: JMP Label.

JMP near Label (chiếm 3 byte),

JMP short Label (chiếm 2 byte)

JMP far Label (chiếm 5 byte)

Dạng2: JMP Target.

JMP AX ; IP←AX

JMP word PTR [AX]; IP←[AX]

JMP dword PTR [AX]; IP←[AX] CS←[AX+2]

Lệnh nhảy có điều kiện J<Điều kiện> có cú pháp:

J<Điều kiện> short_label

Lệnh nhảy có điều kiện trước tiên kiểm tra điều kiện. Nếu điều kiện thoả mãn thì nhảy tới nhãn ***short_label,*** còn không thoả mãn thì lệnh nhảy bị bỏ qua. Lệnh nhảy có điều kiện là lệnh 2 byte, byte đầu là mã lệnh, byte sau là địa chỉ tương đối. Do vậy khoảng cực đại mà nó nhảy được là -128 đến +128. Muốn nhảy xa hơn phải dùng lệnh nhảy không điều kiên.

Lệnh JA (jump if above) có cú pháp:

JA short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ CF = 0 và cờ ZF = 0.

Lệnh JAE (jump if above or equal) có cú pháp:

JAE short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ CF=0.

Lệnh JB (jump if below) có cú pháp:

JB short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ CF=1.

Lệnh JBE (jump if below or equal) có cú pháp:

JBE short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ CF = 1 và cờ ZF = 1.

Lệnh JNA (jump if not above) có cú pháp:

JNA short_label

Chức năng: Tương đương lệnh JBE.

Lệnh JNAE (jump if not above or equal) có cú pháp:

JNAE short_label

Chức năng: Tương đương lệnh JB.

Lệnh JNB (jump if not below) có cú pháp:

JNB short_label

Chức năng: Tương đương lệnh JAE.

Lệnh JNBE (jump if not below or equal) có cú pháp:

JNBE short_label

Chức năng: Tương đương lệnh JA.

Các cậu lệnh JA, JAE, JB, JBE. JNA, JNAE, JNB, JNBE được dùng trong các phép so sánh số không dấu.

Lệnh JG (jump if greater) có cú pháp:

JG short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ SF = OF và cờ ZF = 0.

Lệnh JGE (jump if greater or equal) có cú pháp:

JGE short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ SF=OF.

Lệnh JL (jump if less) có cú pháp:

JL short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ SF<>OF .

Lệnh JLE (jump if less or equal) có cú pháp:

JLE short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ CF<>OF và cờ ZF = 1.

Lệnh JNG (jump if not greater) có cú pháp:

JNG short_label

Chức năng: Tương đương lệnh JLB.

Lệnh JNGE (jump if not greater or equal) có cú pháp:

JNGE short_label

Chức năng: Tương đương lệnh JL.

Lệnh JNL (jump if not less) có cú pháp:

JNG short_label

Chức năng: Tương đương lệnh JGE.

Lệnh JNLE (jump if not les or equal) có cú pháp:

JNLE short_label

Chức năng: Tương đương lệnh JG.

Các lệnh JG, JGE, JL, JLE, JNG, JNL, JNLE được sử dụng trong các phép so sánh số có dấu.

Lệnh JC (jump if carry) có cú pháp:

JC short_label

Chức năng: Tương đương lệnh JB.

Lệnh JNC (jump if not carry) có cú pháp:

JNC short_label

Chức năng: Tương đương lệnh JNB.

Lệnh JZ (jump if zero) có cú pháp:

JZ short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ cờ ZF=1.

Lệnh JE (jump if equal) có cú pháp:

JE short_label

Chức năng: Tương đương lệnh JZ.

Lệnh JNZ (jump if zero) có cú pháp:

JNZ short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ cờ ZF=0.

Lệnh JNE (jump if not equal) có cú pháp:

JNE short_label

Chức năng: Tương đương lệnh JNZ.

Lệnh JS (jump if sign) có cú pháp:

JS short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ cờ SF=1.

Lệnh JNS (jump if not sign) có cú pháp:

JNS short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ cờ SF=0.

Lệnh JO (jump on overflow) có cú pháp:

JO short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ cờ OF=1.

Lệnh JNO (jump if no overflow) có cú pháp:

JNO short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ cờ OF=0.

Lệnh JP (jump on parity) có cú pháp:

JP short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ cờ PF=1.

Lệnh JNP (jump if no parity) có cú pháp:

JNP short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ cờ PF=0.

Lệnh JCXZ (jump if CX zero) có cú pháp:

JCXZ short_label

Chức năng: Nhảy đến nhãn short_label nếu thoả mãn điều kiện CX=0.

3.4.6. Lệnh so sánh có cú pháp

CMP left, right.

Left có thể là thanh ghi hay bộ nhớ còn right có thể là thanh ghi hay bộ nhớ hay hằng số. Có chức năng so sánh toán hạng ***left*** và ***right.*** Kết quả phản ánh trong các cờ trạng thái nhưng không làm thay đổi nội dung toán hạng ***left.***

3.4.7. Nhóm lệnh lặp

Lệnh LOOP có cú pháp:

LOOP short_label

Chức năng: Giảm nội dung CX đi 1 đơn vị và nhảy đến nhãn short_label nếu thoả mãn điều kiện CX<>0. Nếu CX=0 thì lệnh này bị bỏ qua.

Lệnh LOOPE có cú pháp:

LOOPE short_label

Chức năng: Giảm nội dung CX đi 1 đơn vị và nhảy đến nhãn short_label nếu thoả mãn điều kiện cờ ZF=1 và CX<>0. Nếu không thoả mãn thì lệnh này bị bỏ qua.

Lệnh LOOPZ có cú pháp:

LOOPZ short_label

Chức năng: Tương đương lệnh LOOPE.

CALL Label	***CALL far ptr Label***
((SP))← (IP)	((SP))← (CS)
(IP) ← offset Label	((SP)-2)← (IP)
(SP) ← (SP)-2	(IP) ← offset Label
	(CS) ← seg Label
	(SP)← (SP)-4

Lệnh quay lại từ chương trình con có cú pháp:

RET (constant) hay
RETN (constant) hay
RETF (constant).

RETN dùng để kết thúc chương trình con kiểu near, RETF dùng để kết thúc chương trình con kiểu far. Nếu dùng RET thì phụ thuộc vào cách khai báo thủ tục là near thì nó tương đương RETN còn khai báo thủ tục là far thì nó tương đương RETF.

Khi thực hiện lệnh quay trở về thì trình tự tác động sau được thực hiện:

RETN	RETF
(IP)←((SP))	(IP)←((SP))
(SP)← (SP)+2	(CS)←((SP)+2)
	(SP)← (SP)+4

Như vậy, nếu là lệnh quay về gần thì chỉ nội dung của con trỏ chương trình IP được khôi phục từ ngăn xếp nên đỉnh ngăn xếp chỉ tăng lên 2 đơn vị. Còn nếu là lệnh quay về xa thì không những nội dung của con trỏ chương trình IP mà cả nội dung của CS cũng được khôi phục từ ngăn xếp nên đỉnh ngăn xếp tăng lên tới 4 đơn vị. Cặp thanh ghi CS:IP sẽ được gán bằng địa chỉ của lệnh ngay tiếp theo lệnh gọi CALL.

Trường hợp ***RET (n=constant)*** với n chẵn thì nội dung SP được cộng thêm n nhằm loại bỏ một số tham số trong ngăn xếp.

Thí dụ về call near:

```
......
MOV DX, Addr16; gán địa chỉ cổng Addr16 cho DX
MOV AL, Value_8bit; gán giá trị cần chuyển Value_8bit cho AL
CALL Transmitte; gọi chương trình con Transmitte (near)
......
Transmitte PROC near
  OUT DX, AL; xuất dữ liệu trong AL ra cổng có địa chỉ trong DX
  RET   ; quay về trao trả quyền điều khiển cho chương trình chính
Transmitte ENDP
```

Thí dụ về call far:

......

MOV DX, Addr16; *gán địa chỉ cổng Addr16 cho DX*

MOV AL, Value_8bit; *gán giá trị cần chuyển Value_8bit cho AL*

CALL ***far ptr*** Transmitte; *gọi chương trình con Transmitte (far)*

......

Transmitte PROC ***far***

OUT DX, AL; *xuất dữ liệu trong AL ra cổng có địa chỉ trong DX*

RET ; *quay về trao trả quyền điều khiển cho chương trình chính*

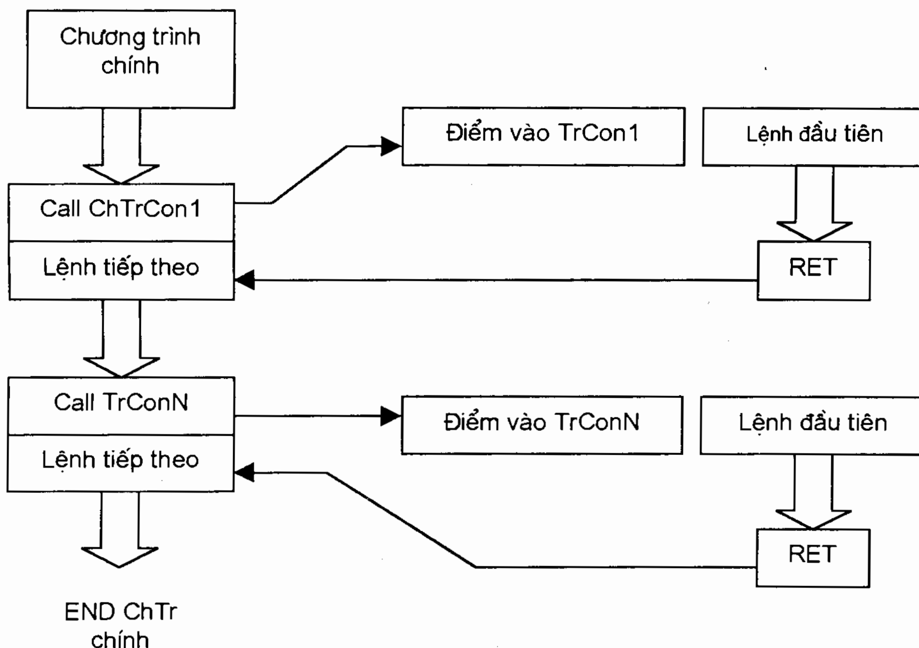
Transmitte ENDP

Dạng ***CALL target*** (reg/mem) cho phép gọi chương trình con có địa chỉ là nội dung của target. Target có thể là toán hạng thanh ghi hay bộ nhớ. Thí dụ:

CALL BX (toán hạng thanh ghi).

CALL word ptr [BX] (toán hạng bộ nhớ – dạng near).

CALL dword ptr [BX] (toán hạng bộ nhớ – dạng far).



Hình 3.2. **Phương thức gọi (CALL) chương trình con**

Hình 3.2 mô tả phương thức gọi chương trình con từ chương trình chính bằng lệnh gọi CALL. Mỗi khi có lời gọi từ chương trình chính thì thao tác cất giữ nội dung con trỏ chương trình vào ngăn xếp được thực hiện và điều khiển được chuyển đến điểm vào của chương trình con tương ứng. Khi kết thúc công việc

của mình, lệnh RET của ***chương trình con cho phép khôi phục lại*** nội dung con trỏ chương trình được cất giữ trước đó nghĩa là trả lại điều khiển cho chương trình chính.

Thông thường ở điểm vào của ***chương trình con là một loạt các thao tác cất giữ nội dung thanh ghi vào ngăn xếp là thông tin trạng thái của chương trình chính bằng các lệnh PUSH các thanh ghi tương ứng. Còn tại điểm ra của chương trình con (trước khi lệnh RET có tác động) là một loạt các thao tác khôi phục nội dung các thanh ghi là thông tin trạng thái của chương trình chính bằng các lệnh POP các thanh ghi tương ứng.***

3.4.9. Nhóm lệnh tính toán số học

Lệnh cộng không nhớ ADD (addition) có cú pháp:

ADD dest, source

Dest←source+ Dest

Chức năng: là phép cộng không nhớ giữa số hạng dest và source, kết quả đặt trong dest. Chú ý là phép cộng không được thực hiện giữa hai thanh ghi mảng và dest chỉ có thể là dạng thanh ghi hay mem.

Lệnh cộng có nhớ ADC (add with carry) có cú pháp:

ADC dest, source

Dest←source+ Dest +(CF)

Chức năng: là phép cộng có nhớ giữa số hạng dest và source, tức là phải cộng với nội dung CF. Kết quả đặt trong dest. Chú ý là phép cộng không được thực hiện giữa hai thanh ghi mảng và dest chỉ có thể là dạng thanh ghi hay mem.

Lệnh tăng INC (increment) có cú pháp:

INC dest (reg/mem)

Dest← Dest +1

Lệnh chỉnh thập phân AAD (ascii adjust for addition) dùng để cộng hai số mã hoá ASCII mà kết quả cho ở dạng thập phân. AL là toán hạng đích.

Lệnh chỉnh thập phân DAA (decimal adjust for addition) dùng để cộng hai số mã hoá nhị phân mà kết quả cho ở dạng thập phân. AL là toán hạng đích.

Lệnh trừ không nhớ SUB (substract) có cú pháp:

SUB dest, source

Dest← Dest- source

Chức năng: là phép trừ không nhớ giữa số bị trừ dest và số trừ source, kết quả đặt trong dest. Chú ý là phép trừ không được thực hiện giữa hai thanh ghi mảng và dest chỉ có thể là dạng *thanh ghi* hay *mem*.

Lệnh trừ có nhớ SBB có cú pháp:

SBB dest, source

Dest← Dest - source - (CF)

Chức năng: là phép trừ có nhớ giữa số số bị trừ dest và số trừ source và trừ di nội dung CF. Kết quả đặt trong dest. Chú ý là phép trừ có nhớ không được thực hiện giữa hai thanh ghi mảng và dest chỉ có thể là dạng *thanh ghi* hay *mem*.

Lệnh giảm DEC (decrement) có cú pháp:

DEC dest (reg/mem)

Dest← Dest –1

Lệnh này chỉ ảnh hưởng tới cờ AF, OF, PF, SF, ZF mà không ảnh hưởng tới cờ CF.

Lệnh NEG có cú pháp:

NEG dest (reg/mem)

Dest← - Dest

Lệnh này dùng để đổi dấu toán hạng đích.

Lệnh chỉnh thập phân AAS (ascii adjust for substract) dùng để trừ hai số mã hoá ASCII mà kết quả cho ở dạng thập phân. AL là toán hạng đích.

Lệnh chỉnh thập phân DAS (decimal adjust for substract) dùng để trừ hai số mã hoá nhị phân mà kết quả cho ở dạng thập phân. AL là toán hạng đích.

Lệnh nhân số không dấu ***MUL*** (multiple) có cú pháp:

MUL source

Chức năng: là phép nhân không dấu. Nếu source là toán hạng 8 bit thì AL được ngầm định chứa thừa số thứ nhất còn source là thừa số thứ hai. Kết quả 16 bit chứa trong AX. Nghĩa là:

MUL source_8bit

(AX) ←(AL)*(source_8bit)

Nếu source là toán hạng 16 bit thì AX được ngầm định chứa thừa số thứ nhất còn source là thừa số thứ hai. Kết quả 32 bit chứa trong cặp DX:AX. Nghĩa là:

MUL source_16bit

(DX:AX) ←(AX)*(source_16bit)

Lệnh nhân số có dấu ***IMUL*** có cú pháp:

IMUL source

Chức năng: Giống như nhân số không dấu MUL nhưng là với toán hạng có dấu. Kết quả cũng là số có dấu. Lệnh này chỉ ảnh hưởng tới cờ CF và OF. Nếu CF=1 và OF=1 thì AH hay DX chứa phần cao của kết quả. Còn nếu CF=0 và OF=0 thì AH hay DX biểu thị dấu của kết quả.

Lệnh hiệu chỉnh thập phân AAM (ascii adjust for multiple) có cú pháp:

AAM

Chức năng: Hiệu chỉnh phép nhân hai số BCD mã hoá ASCII trong AL vào AX.

Lệnh chia số không dấu ***DIV*** (Division) có cú pháp:

DIV source

Chức năng: là phép chia không dấu. Nếu source là toán hạng 8 bit thì AX được ngầm định chứa số bị chia còn source là số chia. Thương số chứa trong AL còn số dư trong AH. Nghĩa là:

DIV source_8bit

(AL) ←(AX) DIV (source_8bit)

(AH) ←(AX) MOD (source_8bit)

Nếu source là toán hạng 16 bit thì DX:AX được ngầm định chứa số bị chia còn source là số chia. Thương số chứa trong AX còn số dư trong DX. Nghĩa là:

(AX) ←(DX:AX) DIV (source_16bit)

(DX) ←(DX:AX) MOD (source_16bit)

Lệnh DIV không ảnh hưởng tới các cờ.

Lệnh chia số có dấu ***IDIV*** có cú pháp:

IDIV source

Chức năng: Giống như chia số không dấu DIV nhưng là với toán hạng có dấu. Kết quả cũng là số có dấu. Lệnh IDIV không ảnh hưởng tới các cờ. Sau phép chia AL chứa thương số (số có dấu) còn AH chứa số dư (số có dấu). Nếu source=0 hoặc nằm ngoài –128..+127 hoặc -32768 → +32767 (phụ thuộc vào source là 8 bit hay 16 bit) thì CPU sẽ tự động gọi ngắt INT0.

Lệnh hiệu chỉnh thập phân AAD (ascii adjust for division) có cú pháp:

AAD

Chức năng: Hiệu chỉnh phép chia hai số ASCII.

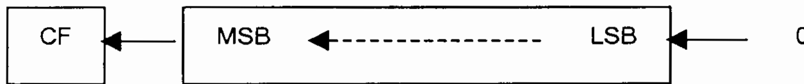
3.4.10. Nhóm lệnh dịch chuyển và quay vòng

Lệnh SHL (shift logical left) có cú pháp:

SHL dest,1 hay SHL dest,CL

Chức năng: Dịch sang trái toán hạng dest 1 bit (trường hợp 1) hay dịch sang trái toán hạng dest với số bước bằng CL bit (trường hợp 2). Bit có trọng số lớn nhất hiện hành →CF còn bit có trọng số nhỏ nhất hiện hành được gán bằng 0 (hình 3.3).

Chú ý: Đối với số không dấu thì dịch trái 1 bit tương đương nhân nó với 2. Các cờ CF, OF, PF, SF, ZF bị tác động khi thực hiện lệnh này.



Hình 3.3. Phương thức dịch chuyển của lệnh SHL

Lệnh SHR (shift logical right) có cú pháp:

SHR dest,1 hay SHR dest,CL

Chức năng: giống như lệnh SHL nhưng diễn ra theo chiều ngược lại.

Lệnh dịch chuyển số học SAL có cú pháp:

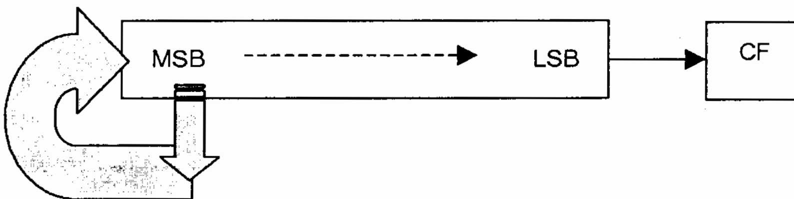
SAL dest,1 hay SAL dest, CL

Chức năng: Giống hoàn toàn lệnh SHL.

Lệnh dịch chuyển số học SAR có cú pháp:

SAR dest, 1 hay SAR dest, CL

Chức năng: Giống hoàn toàn lệnh SHR nhưng bit dấu (bit có trọng số lớn nhất) không thay đổi (hình 3.4).

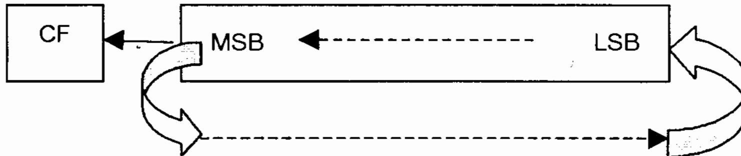


Hình 3.4. Phương thức dịch chuyển của lệnh SAR

Lệnh quay ROL (rotate left) có cú pháp:

ROL dest,1 hay SHL dest,CL

Chức năng: Quay sang trái toán hạng dest 1 bit (trường hợp 1) hay quay sang trái toán hạng dest với số bước bằng CL bit (trường hợp 2). Bit có trọng số lớn nhất hiện hành →CF còn bit có trọng số nhỏ nhất hiện hành được gán bằng bit có trọng số lớn nhất (hình 3.5).

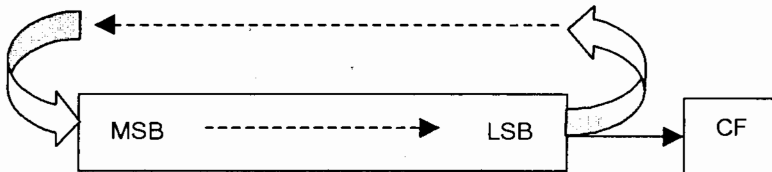


Hình 3.5. Phương thức dịch chuyển của lệnh ROL

Lệnh quay ROR (rotate right) có cú pháp:

ROR dest,1 hay SHR dest,CL

Chức năng: Quay sang phải toán hạng dest 1 bit (trường hợp 1) hay quay sang phải toán hạng dest với số bước bằng CL bit (trường hợp 2). Bit có trọng số nhỏ nhất hiện hành →CF còn bit có trọng số lớn nhất hiện hành được gán bằng bit có trọng số nhỏ nhất (hình 3.6).

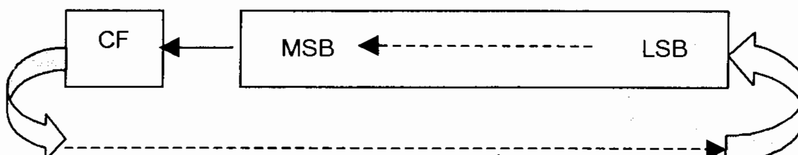


Hình 3.6. Phương thức dịch chuyển của lệnh ROR

Lệnh quay trái qua cờ CF ***RCL*** (rotate through carry left) có cú pháp:

RCL dest,1 hay RCL dest,CL

Chức năng: Quay sang trái toán hạng dest 1 bit (trường hợp 1) hay quay sang trái toán hạng dest với số bước bằng CL bit (trường hợp 2). Bit có trọng số lớn nhất hiện hành → CF còn bit có trọng số nhỏ nhất hiện hành được gán bằng bit CF (hình 3.7).

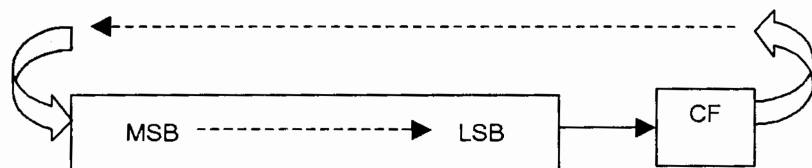


Hình 3.7. Phương thức dịch chuyển của lệnh RCL

Lệnh quay phải qua cờ CF ***RCL*** (rotate through carry right) có cú pháp:

RCR dest,1 hay RCR dest, CL

Chức năng: Giống như lệnh RCL nhưng theo chiều ngược lại (hình 3,8).



Hình 3.8. Phương thức dịch chuyển của lệnh RCR

3.4.11. Nhóm lệnh thực hiện phép tính logic

Lệnh nhân logic AND có cú pháp:

AND dest, source

Dest← dest AND source

CF← 0; OF← 0

Chức năng: Nhân logic giữa toán hạng dest với toán hạng source, kết quả đặt trong dest. Dest phải là dạng reg hay mem. Lệnh này có tác động tới các cờ CF, OF, PF, SF, SF, ZF.

Lệnh cộng logic OR có cú pháp:

OR dest, source

Dest← dest OR source

CF← 0; OF← 0

Chức năng: Cộng logic giữa toán hạng dest với toán hạng source, kết quả đặt trong dest. Dest phải là dạng reg hay mem. Lệnh này có tác động tới các cờ CF, OF, PF, SF, SF, ZF.

Lệnh cộng modul 2 XOR có cú pháp:

XOR dest, source

Dest← dest XOR source

CF← 0; OF← 0

Chức năng: Cộng modul 2 giữa toán hạng dest với toán hạng source, kết quả đặt trong dest. Dest phải là dạng reg hay mem. Lệnh này có tác động tới các cờ CF, OF, PF, SF, SF, ZF.

Lệnh này thường được sử dụng để phát hiện cặp bit khác nhau giữa hai toán hạng hoặc muốn đảo ngược một bit nào đó.

Lệnh đảo NOT có cú pháp:

NOT dest

Dest $\leftarrow \overline{\text{dest}}$

Lệnh kiểm tra TEST có cú pháp:

TEST dest, source

Chức năng: Giống lệnh nhân logic giữa toán hạng dest với toán hạng source, nhưng kết quả không đặt trong dest mà chỉ tác động tới các cờ CF, OF, PF, SF, SF, ZF.

3.4.12. Nhóm lệnh xử lý xâu chuỗi

Lệnh MOVS (move string) có các dạng

MOVSB (chuyển byte),

MOVSW(chuyển word),

MOVS [ES:] dest, [seg:] source

Chức năng: chuyển dữ liệu vùng nhớ với điều kiện SI trỏ tới vùng source DI trỏ tới vùng dest. Mỗi lần chuyển SI, DI tự động tăng 1 đơn vị (nếu là MOVSB và cờ hướng DF=0) hoặc SI, DI tự động tăng 2 đơn vị (nếu là MOVSW và cờ hướng DF=0). Ngược lại, Mỗi lần chuyển SI, DI tự động giảm 1 đơn vị (nếu là MOVSB và cờ hướng DF=1) hoặc SI,DI tự động giảm 2 đơn vị (nếu là MOVSW và cờ hướng DF=1).

Lệnh so sánh xâu chuỗi CMPS (compare string) có các dạng

CMPSB (so sánh xâu chuỗi kiểu byte),

CMPSW(so sánh xâu chuỗi kiểu word),

CMPS [ES:] dest, [seg:] source

Chức năng: so sánh xâu chuỗi dữ liệu 2 vùng nhớ với điều kiện SI trỏ tới vùng source DI trỏ tới vùng dest. Mỗi lần so sánh SI, DI tự động tăng 1 đơn vị (nếu là CMPSB và cờ hướng DF=0) hoặc SI, DI tự động tăng 2 đơn vị (nếu là CMPSW và cờ hướng DF=0). Ngược lại, mỗi lần so sánh SI, DI tự động giảm 1 đơn vị (nếu là CMPSB và cờ hướng DF=1) hoặc SI,DI tự động giảm 2 đơn vị (nếu là CMPSW và cờ hướng DF=1).

Lệnh quét xâu chuỗi SCAS(scan string) có các dạng

SCASB (so sánh xâu chuỗi kiểu byte),

SCASW(so sánh xâu chuỗi kiểu word),

SCAS [ES:] dest, [seg:] source

Chức năng: quét tìm trong xâu chuỗi dữ liệu của vùng nhớ được trỏ bởi ES:DI. Giá trị dò tìm phải đặt trong AL nếu là SCASB hoặc trong AX nếu là SCASW. Mỗi lần quét xong 1 byte (hay word) DI tự động tăng 1 đơn vị (nếu là SCASB và cờ hướng DF=0) hoặc DI tự động tăng 2 đơn vị (nếu là SCASW và cờ hướng DF=0). Ngược lại, mỗi lần quét xong DI tự động giảm 1 đơn vị (nếu là SCASB và cờ hướng DF=1) hoặc DI tự động giảm 2 đơn vị (nếu là SCASW và cờ hướng DF=1).

Lệnh chuyển LODS(load string) có các dạng

LODSB (nạp byte),

LODSW(nạp word),

LODS [seg:] source

Chức năng: nạp 1 byte hay 1 word từ vùng nhớ được trỏ bởi DS:SI vào AL hay AX. Mỗi lần chuyển xong, SI tự động tăng 1 đơn vị (nếu là LODSB và cờ hướng DF=0) hoặc SI tự động tăng 2 đơn vị (nếu là LODSW và cờ hướng DF=0). Ngược lại, Mỗi lần chuyển xong, SI tự động giảm 1 đơn vị (nếu là LODSB và cờ hướng DF=1) hoặc SI tự động giảm 2 đơn vị (nếu là LODSW và cờ hướng DF=1).

Lệnh lưu trữ STOS(load string) có các dạng

STOSB (nạp byte),

STOSW(nạp word),

STOS [seg:] source

Chức năng: lưu trữ 1 byte hay 1 word từ Al hay AX vào vùng nhớ được trỏ bởi DS:SI. Mỗi lần lưu trữ xong, SI tự động tăng 1 đơn vị (nếu là STOSB và cờ hướng DF=0) hoặc SI tự động tăng 2 đơn vị (nếu là STOSW và cờ hướng DF=0). Ngược lại, Mỗi lần lưu trữ xong, SI tự động giảm 1 đơn vị (nếu là STOSB và cờ hướng DF=1) hoặc SI tự động giảm 2 đơn vị (nếu là STOSW và cờ hướng DF=1).

3.5. TỔ CHỨC MACRO

Macro là tập hợp các lệnh assembly có thể xuất hiện nhiều lần trong một chương trình. Mỗi Macro có một tên riêng để chương trình gọi tới được. Đây là cách xây dựng các lệnh có chức năng rộng lớn hơn so với chức năng của 1 lệnh assembly. Mỗi Macro có thể hình dung như lệnh của ngôn ngữ bậc cao.

3.5.1. Định nghĩa một Macro (khung của Macro)

```
Name MACRO [dummy parameter-list]
(Các lệnh assembly nằm ở đây)
ENDM
```

Như vậy MACRO gồm 3 phần:

Phần header dùng để khai báo một macro: Name là tên Macro cần định nghĩa, MACRO là chỉ dẫn cho trình biên dịch biết đây là cấu trúc macro, dummy là các tham số cần cho macro (có thể có cũng có thể không có).

Phần thân macro gồm các lệnh assembly quy định chức năng của macro.

Phần cuối là lệnh directive ENDM báo cho trình biên dịch biết đây là điểm kết thúc của macro.

Ưu điểm của macro:

Có thể truyền tham số cho macro một cách dễ dàng.

Tốc độ thực hiện nhanh hơn chương trình con vì các thao tác cất giữ và khôi phục thông tin trạng thái không cần thực hiện.

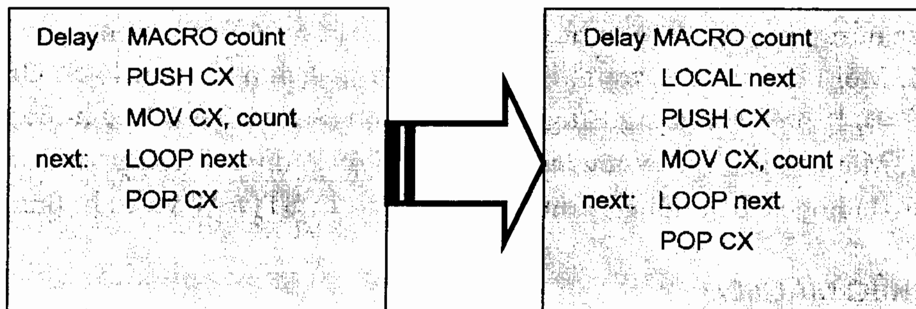
Có thể tạo thư viện các macro một cách dễ dàng.

3.5.2. Các chỉ dẫn (directive) cho Macro

Chỉ dẫn LOCAL có cú pháp

LOCAL localName [,localName] ...

Chức năng: dùng để định nghĩa các ký hiệu chỉ sử dụng riêng cho macro đó. Xét ví dụ tạo macro giữ chậm Delay:



Hình 3.9. **Sử dụng chỉ dẫn LOCAL để tạo nhãn next nhiều lần**

Đối với macro trên chương trình chính chỉ gọi được một lần vì ta đã biết là nhãn ***next*** chỉ được định nghĩa một lần. Muốn gọi được nhiều lần phải dùng chỉ dẫn ***local.*** Chỉ dẫn này báo cho assembly biết để đổi nhãn thành nhãn mới mỗi khi chương trình gọi tới macro (hình 3.9).

Chỉ dẫn điều kiện bao gồm

IFB(if blank) dùng để kiểm tra các tham số truyền cho macro có thiếu không. IFB có cú pháp

IFB <parameter>

(Tại đây các lệnh assemly nếu parameter là trống)

[ELSE (Tại đây các lệnh assemly nếu parameter là không trống)]

ENDIF

IFNB(if not blank) dùng để kiểm tra các tham số truyền cho macro có tồn tại không. IFNB có cú pháp tương tự như IFB.

Chỉ dẫn lặp bao gồm

REPL expression định nghĩa một khối cần lặp với số lần chứa trong ***expression.*** Cấu trúc của nó là:

REPL expression

(Tại đây các lệnh assemly)

ENDM

3.5.3. Các toán tử cho Macro

Toán tử EQ có cú pháp exp1 EQ exp2 sẽ cho kết quả TRUE (=-1) nếu exp1 = exp2.

Toán tử NE có cú pháp exp1 NE exp2 sẽ cho kết quả TRUE (=-1) nếu exp1 <> exp2.

Toán tử LE có cú pháp exp1 LE exp2 sẽ cho kết quả TRUE (=-1) nếu exp1 <= exp2.

Toán tử GT có cú pháp exp1 GT exp2 sẽ cho kết quả TRUE (=-1) nếu exp1 > exp2.

Toán tử GE có cú pháp exp1 GE exp2 sẽ cho kết quả TRUE (=-1) nếu exp1 >= exp2.

3.6. XÂY DỰNG CHƯƠNG TRÌNH ASEMBLY

3.6.1. Các bước xây dựng chương trình

- *Bước 1:* Xây dựng lưu đồ thuật toán tổng quát. Tại đây các yếu tố của nhiệm vụ chương trình được xem xét và phân tích nhằm đưa ra giải pháp và phương thức thực hiện tối ưu. Nếu nhiệm vụ của chương trình là phức tạp thì nó sẽ được tách ra thành các nhiệm vụ con với mối quan

hệ được xác định trong thuật toán tổng quát. Mỗi nhiệm vụ con phải có lưu đồ thuật toán riêng để giải quyết trọn vẹn chức năng của nhiệm vụ con đó.

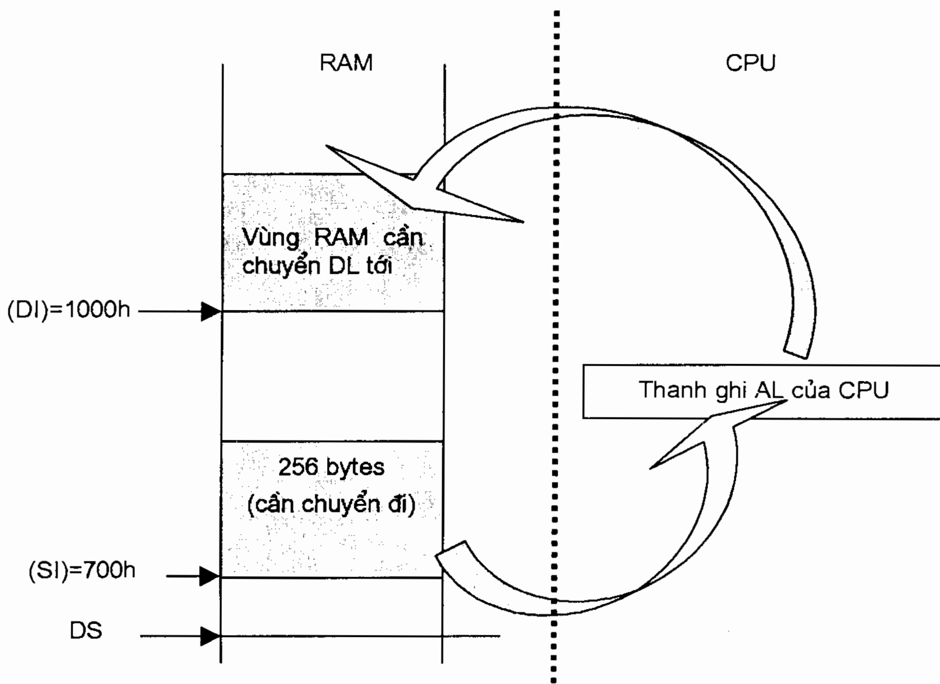
- *Bước 2:* Xây dựng chương trình nguồn assembly. Nếu chức năng không lớn thì chương trình nguồn assembly được viết ở dạng đơn giản, chỉ cần một thủ tục chính. Nếu chức năng lớn hay rất lớn thì chương trình nguồn assembly được viết ở dạng phức tạp hơn về cấu trúc. Có thể là một file gồm nhiều thủ tục, nhiều macro hay chương trình gồm nhiều file khác nhau.

3.6.2. Chương trình minh hoạ

◆ *Chuyển mảng dữ liệu*

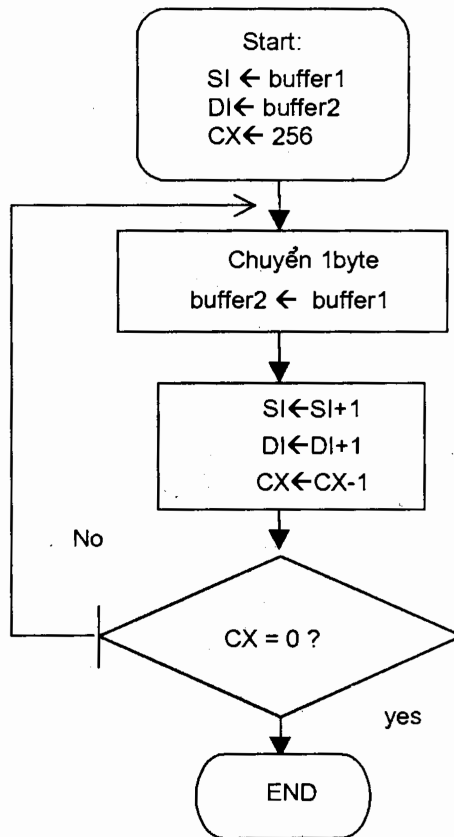
Bài tập 1: Viết chương trình chuyển mảng dữ liệu từ vùng nhớ Buffer1 có địa chỉ bắt đầu là 700h tới vùng nhớ Buffer2 có địa chỉ bắt đầu là 1000h. Biết rằng cả hai vùng nhớ đều nằm trong mảng do thanh ghi DS quản lý và kích thước 2 vùng nhớ bằng nhau và bằng 256 byte.

Giải: Có thể hình dung quá trình chuyển dữ liệu qua sơ đồ hình 3.10.



Hình 3.10. **Sơ đồ mô tả quá trình chuyển dữ liệu của bài tập 1.**

Bước1: Xây dựng lưu đồ thuật toán.



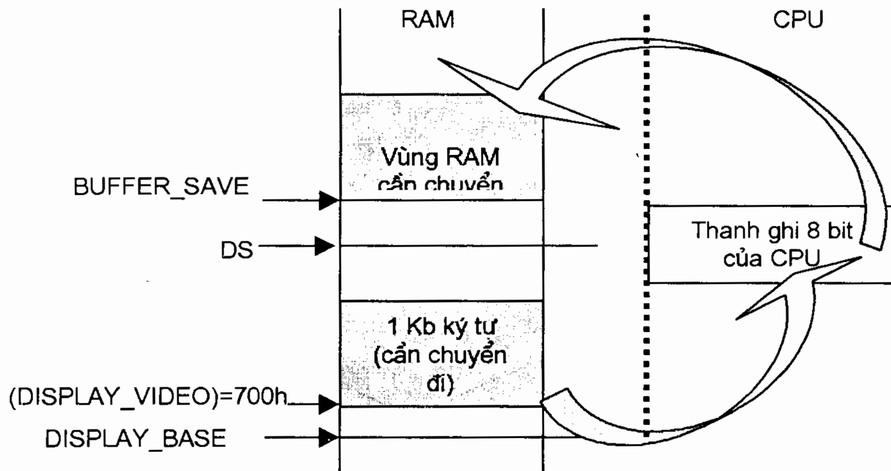
Bước2: Xây dựng chương trình nguồn.

```
Code_Seg SEGMENT; mở mảng lệnh
        ASSUME CS: Code_Seg
        ORG 100h; Tạo file dạng COM
ThuTucChinh PROC; thủ tục chính
        CALL Chuyen_Mang; gọi chương trình con Chuyển_Mảng
ThuTucChinh EndP;
Chuyen_mang PROC ; thủ tục được đặt tên là chuyển_mảng
        MOV SI, 700h; SI trỏ tới offset của Buffer1
        MOV DI, 1000h; SI trỏ tới offset của Buffer2
        MOV CX, 256; CX là bộ đếm 256 byte cần chuyển
Loop256:
        MOV AL,[SI]; chuyển nội dung  ngăn nhớ do SI trỏ tới vào AL
        MOV [DI],AL; chuyển nội dung AL vào ngăn nhớ do DI trỏ tới
        INC SI; tăng SI trỏ tới byte tiếp theo
        INC DI; tăng DI trỏ tới byte tiếp theo
        LOOP Loop256; lặp lại 256 lần
```

Chuyen_mang EndP
Code_Seg EndS ; đóng mảng lệnh
END ThuTucChinh

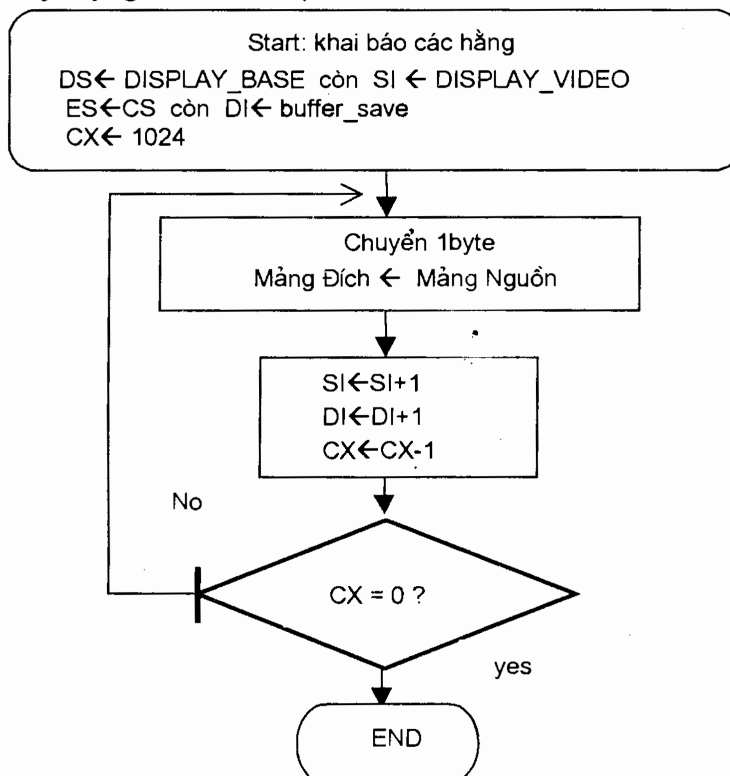
Bài tập 2: Viết chương trình chuyển mảng dữ liệu từ vùng nhớ DISPLAY_VIDEO (địa chỉ 700h) có địa chỉ mảng là DISPLAY_BASE (địa chỉ B800h) tới vùng dữ lệu BUFFER_SAVE có địa chỉ mảng do DS quản lý. Biết rằng vùng nhớ DISPLAY_VIDEO chứa 1 Kb ký tự (mã hoá ASCII).

Giải: Có thể hình dung quá trình chuyển dữ liệu qua sơ đồ hình 3.11.



Hình 3.11. **Sơ đồ mô tả quá trình chuyển dữ liệu của bài tập 2.**

Bước1: Xây dựng lưu đồ thuật toán.



Bước2: Trong chương trình này sẽ sử dụng lệnh MOVSB thay vì dùng lệnh MOV bình thường. Chương trình nguồn có dạng:

```
;-------------------------------------------
;--------Định nghĩa các hằng số---------
;-------------------------------------------
        DISPLAY_BASE  EQU 0B800h
        DISPLAY_VIDEO EQU  700h
;-------------------------------------------
Code_Seg        SEGMENT; mở mảng lệnh
ASSUME CS: Code_Seg, DS: Data_Seg
  ORG 100h;                     Tạo file dạng COM
   ThuTucChinh PROC              ; thủ tục chính
  CALL Chuyen_Mang         ; gọi chương trình con Chuyển_Mảng
  ThuTucChinh EndP;
;-------------------------------------------
Chuyen_Mang PROC NEAR
  PUSH AX
  PUSH BX
  PUSH CX
  MOV SI, DISPLAY_VIDEO; SI trỏ tới vùng chứa ký tự cần chuyển
  LEA DI,BUFFER_SAVE; DI trỏ tới vùng sẽ chuyển ký tự tới
  MOV AX,DISPLAY_BASE
  MOV DS, AX ;    DS:SI trỏ tới vùng chứa ký tự cần chuyển đi
  MOV AX, CS
  MOV ES, AX;     ES:DI trỏ tới vùng sẽ chuyển ký tự tới
  CLD  ; xoá cờ DF: DF =0
  MOV CX, 1024  ; Bộ đếm 1024
LOOP_1Kb:
  MOVSB       ; dùng lệnh chuyển xâu ký tự kiểu byte
LOOP LOOP_1Kb ; lặp lại cho tới hết 1024 byte
  POP CX
  POP BX
  POP AX
  RET
```

```
Chuyen_Mang ENDP
;------------------------
Code_Seg ENDS                      ; đóng mảng lệnh
;------------------------------------------
Data_Seg SEGMENT                   ; mở mảng dữ liệu
   BUFFER_SAVE  DB 1024 DUP(?); Cấp phát vùng nhớ để chứa dữ liệu
Data_Seg EndS                      ; đóng mảng dữ liệu
;-------------------------------------------
END ThuTucChinh
```

Điểm chú ý ở đây là hai vùng nhớ cần chuyển dữ liệu nằm ở hai mảng khác nhau nên mỗi vùng phải sử dụng con trỏ đầy đủ (*seg: offset*). Mặt khác, ta muốn dùng lệnh MOVSB thay vì lệnh MOV để chuyển ký tự nên phải chuẩn bị trước cặp DS:SI cho trỏ tới vùng dữ liệu nguồn DISPLAY_VIDEO còn cặp ES:DI cho trỏ tới vùng dữ liệu đích BUFFER_SAVE.

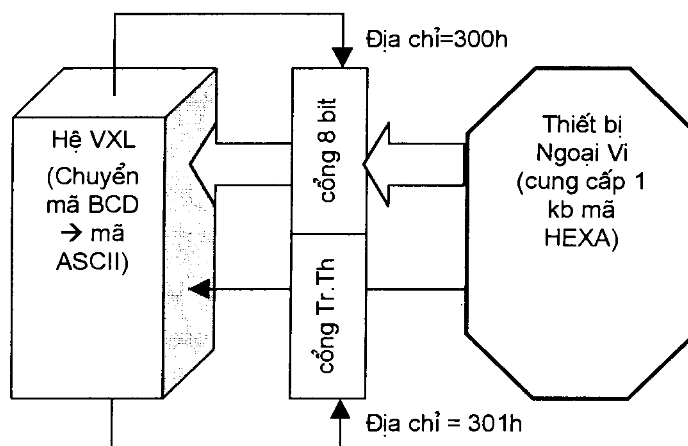
♦ *Chuyển đổi mã*

Bài tập 1: Viết chương trình chuyển dữ liệu dạng mã Hex từ cổng 8 bit có địa chỉ 300h tới vùng BufferRam sau khi đã đổi thành mã ASCII. Biết rằng

a) số lượng byte Hex nhận được là 1024 byte,

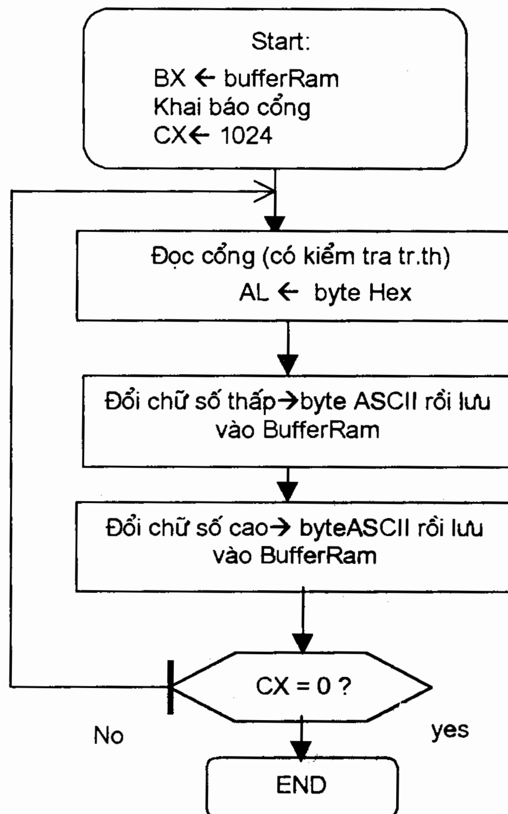
b) cổng trạng thái 8 bit có địa chỉ 301h có chức năng xác nhận trạng thái có dữ liêu: bit D0 =1 báo là có dữ liệu, ngược lại D0=0 báo không có dữ liệu.

Giải: Căn cứ vào yêu cầu đầu bài ta có sơ đồ hoạt động của hệ vi xử lý như hình 3.12.



Hình 3.12. **Sơ đồ hệ đổi mã BCD→ASCII của bài tập 1.**

Bước1: Xây dựng lưu đồ thuật toán.



Bước 2: Xây dựng chương trình nguồn: trong chương trình này sẽ sử dụng từ khoá GROUP để gom các mảng khác như Code_Seg và Data_Seg vào một mảng chung có tên là Cgroup. Đây là dạng thức thường gặp khi xây dựng các chương trình dạng COM.

```
Cgroup GROUP Code_Seg, Data_Seg
ASSUME CS: Cgroup, DS:Cgroup
;---------------------------------------------
;--------Định nghĩa các hằng số---------
;---------------------------------------------
PortData EQU    300h;
            PortStatus EQU  301h
Code_Seg        SEGMENT
ORG 100h
;---------------------------------------------
    ThuTucChinh PROC                ; chương trình chính
  CALL HEX_ASCII_COVERT; gọi chương trình con
  ThuTucChinh EndP;
```

```
;-------------------------------------------
HEX_ASCII_COVERT PROC
  PUSH AX
  PUSH BX
  PUSH CX
  MOV BX, Offset BufferRAM; BX trỏ tới vùng lưu kết quả
  MOV CX, 1024;
Loop_1Kb:
  CALL InPort              ; gọi thủ tục nhận byte Hex từ cổng
  MOV AH,AL                ; gửi tạm sang AH
  AND AL,0FH               ; lọc bỏ 4 bit cao
  CALL CREATE_ ASCII _DIGIT; gọi thủ tục đổi mã→ascii
  MOV AL,AH                ;nhận lại giá trị cũ
  PUSH CX
  MOV CX,4                 ; bộ đếm bước dịch CX=4
  SHR AL,CL                ; dịch AL sang phải 4 bit
  POP CX
CALL CREATE_ASCII_DIGIT; gọi thủ tục đổi mã→ascii
LOOP    Loop_1Kb           ;lặp lại 1024 lần
  POP CX
  POP BX
  POP AX
  RET
HEX_ASCII_COVERT ENDP
;-------------------------------------------
CREATE_ ASCII _DIGIT PROC NEAR
  PUSH AX
  CMP AL,10; so sánh với 10
  JAE HEX_LETTER; nếu >=10 thì là vùng chữ số ABCDEF
  ADD AL,'0'; nếu <10 thì là vùng chữ số 0123456789+ghép với '0'
  JMP  SAVE_BYTE_DIGIT  ; gọi thủ tục SAVE_BYTE_DIGIT để
                        ; lưu trữ vào BufferRam
```

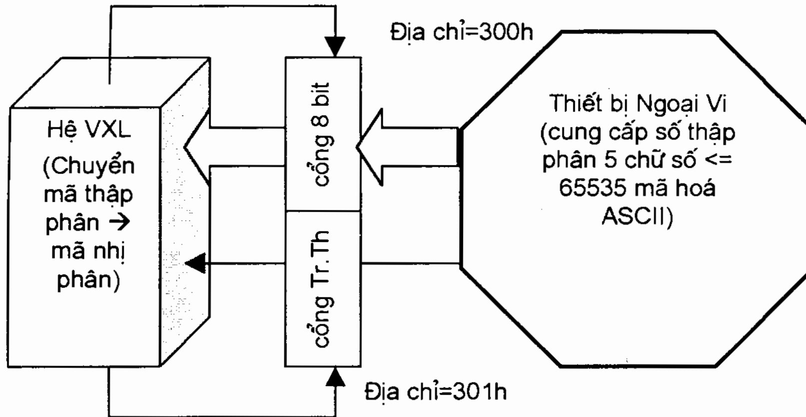
```
HEX_LETTER:
  ADD AL,'A'-10      ; là vùng chữ số ABCDEF thì ghép với 'A'-10
SAVE_BYTE_DIGIT:
   MOV [BX],AL      ; lưu trữ vào BufferRam
   INC BX;  tăng con trỏ BX lên 1 đơn vị
    POP AX
     RET
CREATE_ ASCII _DIGIT ENDP
;--------------------------------
InPort PROC NEAR
  PUSH DX
 MOV DX,PortStatus
 Read_Again:
  IN AL,DX           ;đọc cổng trạng thái
  AND AL, 01h        ; lọc lấy bít DO
  JZ Read_Again      ; DO=0 →cờ zero bật lên →chưa có dữ liệu
MOV DX, PortData
  IN AL, DX          ;  nếu có dữ liệu thì đọc cổng dữ liệu
  POP DX
  RET
InPort EndP
;------------------------------
Code_Seg EndS
Data_Seg SEGMENT
   BufferRAM  db 2*1024 DUP(0) ; cấp phát 2 Kb cho vùng nhớ kết quả
Data_Seg EndS
END ThuTucChinh
```

Như vậy, mỗi byte Hex nhận được từ cổng 300h sẽ được đổi thành 2 byte ASCII rồi được cất giữ vào vùng BufferRam có dung lượng 2x1024 byte. Mỗi byte Hex nhận được, được thủ tục HEX_ASCII_COVERT tách đôi rồi ghép từng giá trị 4 bit với gốc mã ASCII để tạo thành byte ASCII.

Bài tập 2: Viết chương trình chuyển số thập phân 5 chữ số <=65.535 (mỗi chữ số được mã hoá ASCII) và được đưa vào hệ vi xử lý lần lượt từ chữ số cao tới chữ số thấp qua cổngvào 8 bit có địa chỉ 300h thành giá trị nhị phân chứa trong biến bộ nhớ. Biết rằng cổng trạng thái 8 bit có địa chỉ 301h có chức năng

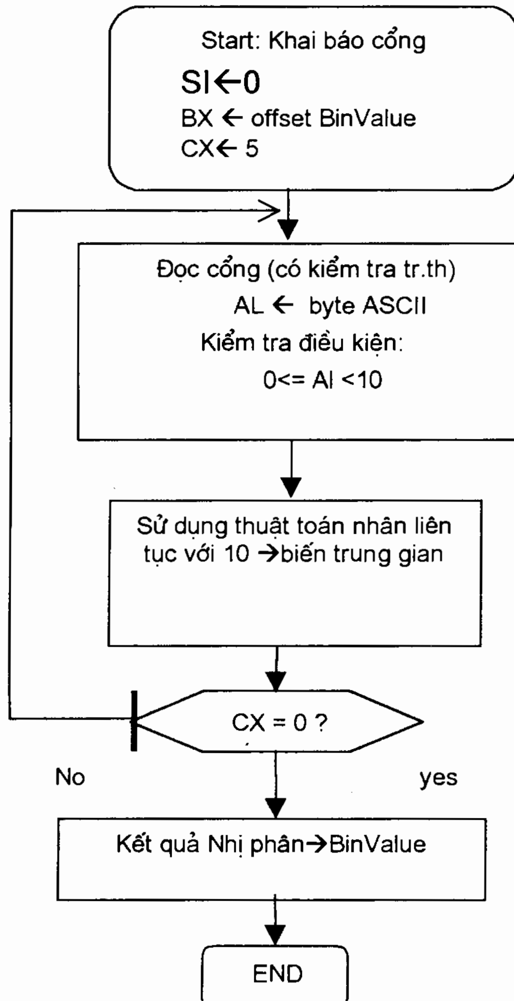
xác nhận trạng thái có dữ liêu: bit D0 =1 báo là có dữ liệu, ngược lại D0=0 báo không có dữ liệu.

Giải: Căn cứ vào yêu cầu đầu bài ta có sơ đồ hoạt động của hệ vi xử lý như hình 3.13.



Hình 3.13. **Sơ đồ hệ đổi mã thập phân mã hoá ASCII→Nhị phân của bài tập 2.**

Bước1: Xây dựng lưu đồ thuật toán.



Bước2: Xây dựng chương trình nguồn.

```
Cgroup GROUP Code_Seg, Data_Seg
 ASSUME CS: Cgroup, DS:Cgroup
;--------------------------------------------
;--------Định nghĩa các hằng số----------
;--------------------------------------------
PortData EQU   300h;
          PortStatus EQU  301h
Code_Seg      SEGMENT
ORG 100h
;--------------------------------------------
   ThuTucChinh PROC                ; thủ tục chính
  CALL ASCII_BIN_COVERT  ; gọi chương trình con
  ThuTucChinh EndP;
;--------------------------------
ASCII_BIN_COVERT PROC
 PUSH AX
 PUSH BX
 PUSH CX
 PUSH DX
 PUSH SI
 PUSH DI
 MOV SI,0; SI chứa tạm kết quả trung gian
 MOV CX, 5;  bộ đếm số lượng chữ số thập phân CX=4
 MOV BX, 0  ;BX chứa số mới nhập cho mỗi lần xử lý
 MOV DI, 10; số nhân cho thuật toán nhân 10
Loop_IN:
 CALL InPort             ; gọi thủ tục nhận byte ASCII từ cổng
 CMP AL,30h          ;<'0' không phải số thập phân
 JC Loop_IN               ;quay lại
 CMP AL,3Ah          ;>='A' cũng không phải số thập phân
 JNC Loop_IN               ;quay lại
 SUB AL,30h  ;là số thập phân thì bỏ đuôi mã ASCII
 MOV BL,AL; gửi tạm sang BL
 MOV AX,SI
```

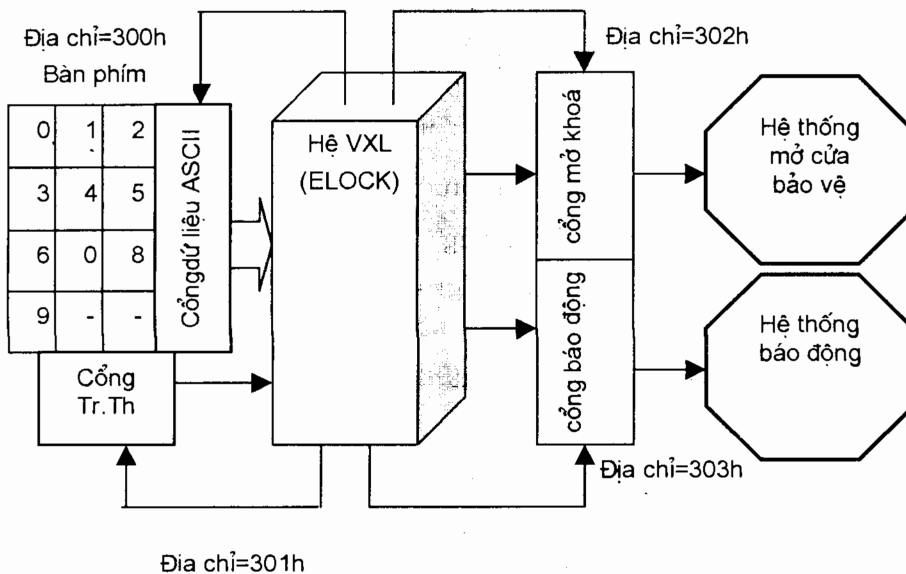
```
    MUL DI;    Nhân giá trị cũ với 10
    ADD AX,B X; cộng với giá trị chữ số mới nhập
    MOV SI,AX
    LOOP Loop_IN; chưa hết 5 chữ số thì quay lại tiếp tục
    MOV BX, OFFSET BinValue; cho BX trỏ tới biến BinValue
    MOV WORD PTR [BX], SI; chuyển vào biến BinValue
    POP DI
    POP SI
    POP DX
    POP CX
    POP BX
    POP AX
    RET
ASCII_BIN_COVERT ENDP
;-----------------------------------------
InPort PROC NEAR
    PUSH DX
  MOV DX,PortStatus
  Read_Again:
    IN AL,DX          ;đọc cổng trạng thái
    AND AL, 01h       ; lọc lấy bít DO
    JZ Read_Again     ; DO=0 →cờ zero bật lên →chưa có dữ liệu
  MOV DX, PortData
    IN AL, DX         ; nếu có dữ liệu thì đọc cổng dữ liệu
    POP DX
    RET
InPort EndP
;-------------
Code_Seg ENDS
;-----------------------------
Data_Seg SEGMENT
  BinValue  dw 0; biến bộ nhớ 2 byte
Data_Seg EndS
END ThuTucChinh
```

Như vậy, thuật toán nhân liên tục với 10 được áp dụng để chuyển dần các chữ số thập phân từ cao đến thấp sang trái trước khi nhận số mới. Số lượng chữ số thập phân trong trường hợp này là 5 với giá trị<=65.535 để chứa đủ trong biến 2 byte BinValue.

♦ ***Điều khiển***

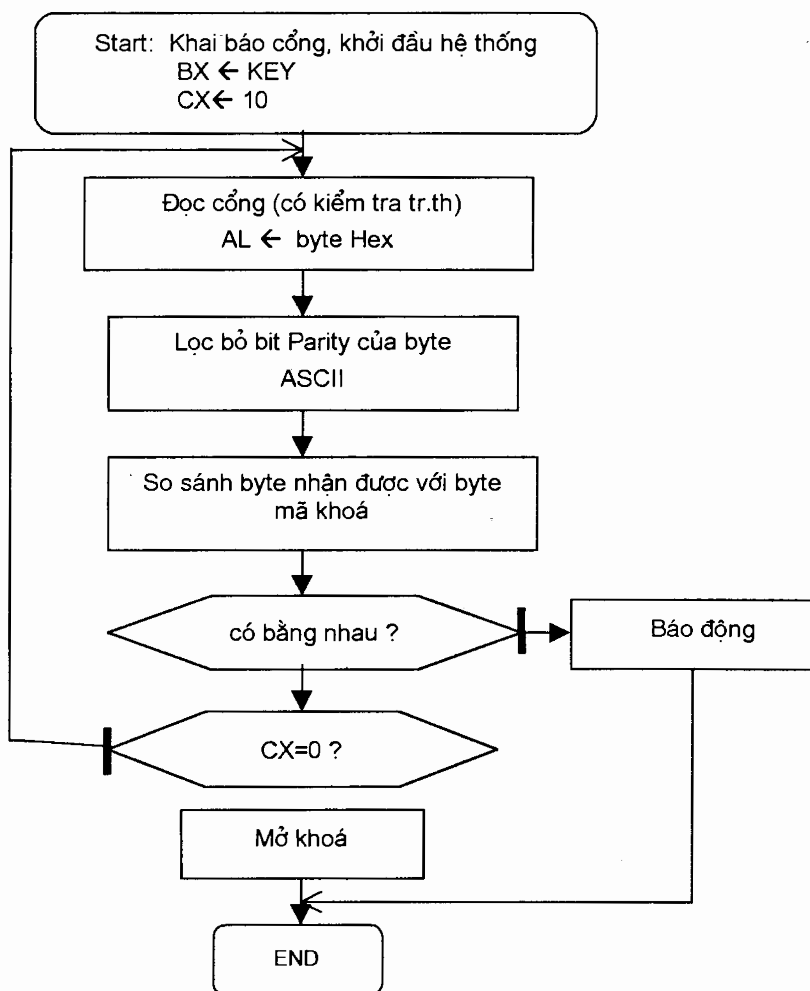
Bài tập 1: Viết chương trình điều khiển chức năng hệ Vi xử lý điều khiển khoá số tự động. Bảng mã số 10 byte được cài trong bộ nhớ của hệ Vi xử lý, mỗi byte được mã hoá ASCII. Một bàn phím của khoá điện tử được mã hoá ASCII đủ (có 8 bit) để người sử dụng thực hiện thao tác mở khoá. Khi mở khoá phải ấn số theo trình tự đúng và đưa qua cổng 300h thì khoá sẽ mở bằng bit DO của cổng 302h. Trong trường hợp không đúng thì hệ vi xử lý sẽ báo động qua bit DO của cổng 303h. Biết rằng cổng trạng thái 8 bit có địa chỉ 301h có chức năng xác nhận trạng thái phím bị ấn: bit D0 =1 báo là có dữ liệu, ngược lại D0=0 báo không có dữ liệu.

Giải: Căn cứ vào yêu cầu đầu bài ta có sơ đồ hoạt động của hệ vi xử lý thực hiện chức năng khoá điện tử như hình 3.14.



Hình 3.14. **Sơ đồ hệ vi xử lý ELOCK**

Bước1: Xây dựng lưu đồ thuật toán.



Bước2: Xây dựng chương trình nguồn.

```
Cgroup GROUP Code_Seg, Data_Seg
        ASSUME CS: Cgroup,DS:Cgroup
;-------------------------------------------
;--------Định nghĩa các hằng số---------
;-------------------------------------------
        PortData        EQU  300h;
        PortStatus      EQU  301h
        LOCK_OPEN    EQU  302h
        LOCK_PIC       EQU  303h
Code_Seg        SEGMENT
        ORG 100h
ThuTucChinh PROC              ; thủ tục chính
        CALL ELOCK; gọi chương trình con
```

```
ThuTucChinh EndP;
;--------------------------------
ELOCK PROC
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        MOV AL,00H            ;cho bit điều khiển d0=0
        MOV DX, LOCK_OPEN  ;địa chỉ cổng mở khoá
        OUT DX,AL             ;đưa ra cổng mở khoá
        MOV DX,LOCK_PIC       ;địa chỉ cổng báo động
        OUT DX,AL          ;đưa ra cổng báo động
        MOV BX, Offset KEY    ; địa chỉ bảng mã khoá
        MOV CX, 10            ; CX là bộ đếm byte mã khóa
NEXT:
        CALL InPort        ; gọi thủ tục nhận byte Hex từ cổng
        AND AL,7FH           ; lọc bỏ đi bit parity
        CMP AL,[BX]          ; so sánh với mã khoá
        JNZ PIC              ; không đúng thì báo động
        INC BX              ; tăng con trỏ bảng mã khoá
        LOOP NEXT        ; lặp cho hết 10 mã khoá
OPENIT:
        MOV DX,LOCK_OPEN
        MOV AL,01h
        OUT DX,AL  ; lệnh mở khoá
        JMP End_      ; kết thúc
PIC: MOV DX,LOCK_PIC
        MOV AL,01h
        OUT DX,AL    ; lệnh báo động
End_:
        POP CX
        POP BX
        POP AX
        RET
```

```
ELOCK ENDP
;-----------------------------------------
InPort PROC NEAR
        PUSH DX
 MOV DX,PortStatus
 Read_Again:
        IN AL,DX         ;đọc cổng trạng thái
        AND AL, 01h      ; lọc lấy bít DO
        JZ Read_Again    ; DO=0 →cờ zero bật lên →chưa có dữ liệu
  MOV DX, PortData
        IN AL, DX        ;  nếu có dữ liệu thì đọc cổng dữ liệu
        POP DX
        RET
InPort EndP
;---------------------------------------------
Code_Seg ENDS
;---------------------------------------------
Data_Seg SEGMENT
 ;############################################
 ; Lập bảng mã khoá KEY gồm 10 byte theo quy luật trình tự nào đó
 ;############################################
KEY    DB  30H      ; byte mã thứ nhất
       DB  31H      ; byte mã thứ hai
       DB  32H      ; byte mã thứ ba
       DB  33H      ; byte mã thứ tư
       DB  34H      ; byte mã thứ năm
       DB  35H      ; byte mã thứ sáu
       DB  36H      ; byte mã thứ bảy
       DB  37H      ; byte mã thứ tám
       DB  38H      ; byte mã thứ chín
       DB  39H      ; byte mã thứ mười
Data_Seg EndS
END ThuTucChinh
```

Khi bắt đầu hoạt động, hệ vi xử lý sẽ liên tục đọc cổng PortStatus (qua việc xét bit D0) để kiểm tra xem có người ấn số trên bàn phím hay không. Nếu có thì đọc giá trị đó qua cổng PortData. Vì các phím của khoá điện tử được mã hoá ASCII đủ (có 8 bit) nên cần lọc bỏ bit parity là bit thứ tám của byte (bit D7) trước khi so sánh nó với bảng mã khóa được khai báo trong bộ nhớ trung tâm của hệ vi xử lý. Nếu phép so sánh là thoả mãn cả về số lượng (đủ 10 byte mã) cả về giá trị từng cặp byte (chúng phải bằng nhau) thì hệ vi xử lý sẽ mở khoá thông qua cổng LOCK_OPEN. Ngược lại, hệ vi xử lý sẽ báo động thông qua cổng LOCK_PIC.

CHƯƠNG 4
THIẾT KẾ HỆ VI XỬ LÝ CHUYÊN DỤNG

Phải nói ngay rằng bài toán thiết kế hệ vi xử lý chuyên dụng không đồng nhất với thiết kế máy tính PC. Nếu máy tính là do các hãng sản xuất lớn như IBM PC, Compac ...phải tuân theo các ***chuẩn mực thiết kế*** nhằm phổ cập hoá máy tính PC và có thể tương thích với nhau trong quá trình hoạt động đặc biệt khi tổ chức thành mạng máy tính PC kiểu LAN hay WAN thì khi thiết kế các hệ vi xử lý chuyên dụng, người thiết kế có quyền thiết kế các chức năng của hệ chỉ theo yêu cầu cụ thể của nhiệm vụ và có thể thực hiện tất cả các bước từ khâu tổ chức phần cứng của hệ đến khâu xây dựng và cài đặt phần mềm vào hệ rồi hiệu chỉnh hệ thống cho tới khi đạt yêu cầu đề ra. Chính điều đó cho phép người thiết kế chủ động hoàn toàn trong xây dựng và tổ chức hệ vi xử lý chức năng.

Chính vì khả năng đó nên thiết kế hệ vi xử lý chuyên dụng là nội dung quan trọng của môn học Kỹ thuật Vi xử lý. Nội dung này cho phép tạo ra các hệ điều khiển thông minh, các hệ xử lý tin tự động theo yêu cầu và theo nhiệm vụ. Những hệ vi xử lý như thế có những ưu điểm sau:

- Có tính mềm dẻo cao trong thao tác;
- Có tốc độ cao so với hệ đa dụng do chức năng được chuyên năng hoá cao, các thao tác thừa được loại bỏ;
- Có độ tin cậy làm việc cao do các mạch vi điện tử IC sử dụng trong hệ là các IC có mức tổ hợp cao LSI hoặc cực cao VLSI;
- Có thể dễ dàng thay đổi thay đổi thông số, trình tự vận hành kể cả thay đổi chức năng của hệ thống chỉ bằng cách thay đổi phần mềm cài đặt bên trong hệ thống mà không phải thay đổi phần cứng của hệ. Ưu điểm này có lẽ là ưu điểm lớn nhất của hệ vi xử lý, vì vậy hệ vi xử lý trở thành phổ cập và nó có mặt ở hầu hết các lĩnh vực kỹ thuật hiện đại.

4.1. TRÌNH TỰ THIẾT KẾ CÁC HỆ VI XỬ LÝ CHUYÊN DỤNG

Khi thiết kế hệ vi xử lý cần tuân thủ các bước theo trình tự sau:

Bước 1. Phân tích chức năng, nhiệm vụ hệ vi xử lý cần thiết kế

Nhiệm vụ của hệ vi xử lý cần thiết kế bao giờ cũng được mô tả đầy đủ bằng các ngôn ngữ kỹ thuật thích hợp. Người thiết kế phải tiến hành nghiên cứu và phân tích một cách kỹ lưỡng nhiệm vụ và các chức năng chính của hệ vi xử lý. Tìm hiểu và nghiên cứu môi trường làm việc của hệ, đối tượng điều khiển của hệ, đặc trưng và tham số của nguồn thông tin mà hệ cần thu nhận và xử lý v.v... Trên cơ sở của bước phân tích phải phân chia một cách hợp lý chức năng nào thuộc phần cứng đảm nhiệm và những chức năng nào do phần mềm đảm nhiệm.

Phần cứng, trước hết là những thành phần không thể thay thế bằng phần mềm như các cấu trúc vật lý của hệ: bộ nhớ trung tâm, các cổng vào/ra, CPU, kênh hệ thống ...và một số chức năng của phần mềm mà muốn cải thiện tính năng nào đó như về tốc độ chẳng hạn khi phải thực hiện các phép tính với các hàm toán học phức tạp.

Phần mềm đảm nhiệm các chức năng còn lại của hệ thống để thực hiện các thao tác phức tạp đòi hỏi phải xử lý tình huống, ra quyết định trong các điều kiện và yếu tố tác động vào hệ là các điều kiện và yếu tố động (luôn biến đổi).

Bước 2. Tổ chức phần cứng cho hệ vi xử lý cần thiết kế

Trên cơ sở của các phân tích của bước 1, ở bước 2 phải tổ chức được phần cứng của hệ thống. Các nội dung cần thực hiện ở bước này là:

- Xây dựng sơ đồ khối của hệ vi xử lý cần thiết kế, trong đó mỗi thành phần được thể hiện bằng một hộp chức năng. Các thành phần này liên hệ với nhau thông qua kênh hệ thống của hệ, đó là kênh địa chỉ, kênh dữ liệu và kênh điều khiển.

- Tiến hành lựa chọn các chip IC phù hợp với chức năng nhiệm vụ của từng thành phần. Quan trọng nhất là chọn chủng loại bộ vi xử lý (CPU) vì nó sẽ quyết định tới khả năng hoạt động chung của hệ như tốc độ, độ rộng kênh dữ tiệu, khả năng quản lý không gian nhớ và quản lý các thiết bị ngoại vi. Kế đến là các chip bộ nhớ sao cho đủ dung lượng và đúng tính năng của bộ nhớ trung tâm cần tổ chức. Nếu hệ có số lượng ngoại vi lớn phải tổ chức các bộ đệm kênh để tránh ảnh hưởng tới tham số của kênh hệ thống như hiện tượng quá tải kênh. Các bộ đệm kênh hệ thống nên sử dụng dạng đệm 3 trạng thái vì nó cho phép bảo vệ kênh tốt nhất. Trong một số trường hợp cụ thể, khi số lượng ngoại vi ít, có thể nối thẳng không qua bộ đệm kênh nhằm tối giản cấu trúc phần cứng.

- Xây dựng các sơ đồ nối ghép chi tiết cho từng thành phần có tính tới cơ chế điều khiển và cơ chế đồng bộ sao cho thật tối ưu nhằm tránh hiện tượng xung đột trạng thái của hệ. Cụ thể:

Tổ chức bộ nhớ trung tâm có dung lượng và cơ chế xuất nhập thông tin đúng theo yêu cầu. Không gian nhớ mà các chip IC bộ nhớ có thể không chiếm hết không gian nhớ mà hệ có thể quản lý nên cần phân bố cho hợp lý. Khi đã phân bố xong, cần thiết lập bảng địa chỉ của bộ nhớ trung tâm cho thật cụ thể vì nó là thông số quan trọng cho bước xây dựng phần mềm. Cần lưu ý khi tổ chức tín hiệu kích hoạt chip (Chip Select) và điều khiển hướng truyền thông tin sao cho phải đơn trị không trùng lặp.

Tổ chức các ngoại vi cần thiết theo đúng tính năng của từng ngoại vi. Khi thiết kế các ngoại vi này điều quan tâm hơn cả là khả năng và phương thức giao tiếp của chúng với hệ vi xử lý. Có các phương thức giao tiếp có thể sử dụng là: hỏi vòng, ngắt hoặc phương thức truy nhập trực tiếp. Mỗi phương thức có ưu nhược điểm riêng nên người thiết kế phải biết lựa chọn cho đúng với điều kiện và nhiệm vụ cụ thể của hệ mà sử dụng phương thức này hay phương thức khác.

Tổ chức bộ giải mã địa chỉ chọn chip, tức là gán cho mỗi thành phần một địa chỉ riêng biệt mà bộ vi xử lý sẽ sử dụng để kích hoạt các thành phần thuộc quyền quản lý của mình nhằm thực hiện các thao tác cần thiết.

Nhiệm vụ cuối cùng là tổ chức ghép nối tất cả các thành phần với nhau thông qua kênh dữ liệu, địa chỉ và hệ thống để tạo thành hệ hoàn chỉnh, sẵn sàng đi vào hoạt động khi có chương trình MONITOR cài đặt bên trong hệ vi xử lý.

Bước 3. Xây dựng phần mềm cho hệ vi xử lý cần thiết kế.

Trên cơ sở của các phân tích của bước 1, bước 2, ở bước 3 phải xây dựng phần mềm của hệ thống. Các nội dung cần thực hiện ở bước này là:

- Xây dựng thuật toán điều khiển bảo đảm quản lý và phối hợp các chức năng của hệ thật tối ưu như phân phối mức ưu tiên của các chức năng, chỉ rõ các giới hạn của từng chức năng nhằm làm cơ sở để tổ chức lưu đồ thuật toán ở bước kế tiếp.
- Xây dựng lưu đồ thuật toán tổng quát cho hoạt động của hệ vi xử lý. Kế đến là các lưu đồ thuật toán cho các modul chức năng nhằm cụ thể hoá một cách chi tiết các trình tự thao tác của hệ thống.
- Viết chương trình nguồn. Thường thì nên viết dưới dạng sao cho chương trình chạy có dạng file COM là dạng gói gọn trong mảng nhớ <=64 kb. Dạng file này này cho phép chương trình MONITOR phản ứng nhanh với các tác động và yêu cầu.

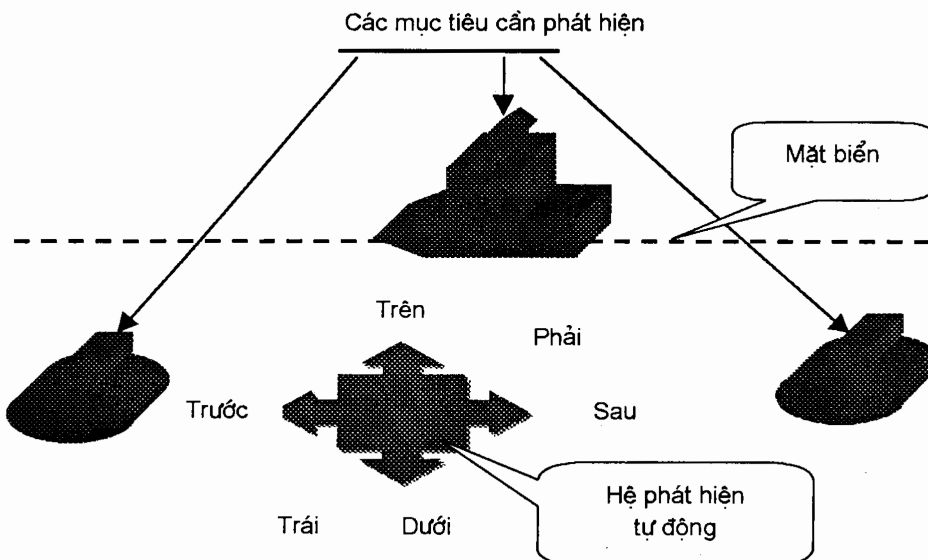
Bước 4. Nạp chương trình cho hệ vi xử lý cần thiết kế

Trên cơ sở của các phân tích của bước 1, bước 2 và bước 3, ở bước 4 phải nạp chương trình cho hệ vi xử lý cần thiết kế theo các cách sau:

- Nếu có hệ vi xử lý phát triển (là thiết bị chuyên dụng cho thiết kế hệ vi xử lý) thì tiến hành mô phỏng chương trình đã xây dựng ở bước 3 cho hệ vi xử lý phát triển. Khi đó, mọi trạng thái của quá trình hoạt động của hệ vi xử lý được ghi nhận và thông báo. Người thiết kế căn cứ vào đó để hiệu chỉnh cả phần cứng và phần mềm cho thật chính xác.
- Nếu không có hệ vi xử lý phát triển thì sử dụng máy nạp ROM để nạp thẳng chương trình đã biên dịch vào ROM của hệ vi xử lý và cho chạy. Quá trình chạy thử người thiết kế phải theo dõi để tiến hành điều chỉnh cho tới khi hệ hoạt động hoàn toàn thoả mãn các yêu cầu đặt ra. Quá trình nạp ROM có thể lặp lại nhiều lần vì mỗi lần sửa chương trình lại phải dịch và nạp lại.

4.2. THIẾT KẾ CÁC HỆ VI XỬ LÝ CHUYÊN DỤNG

4.2.1. Mô tả chức năng hệ vi xử lý cần thiết kế



Hình 4.1. **Hệ thu tin ngẫu nhiên đa kênh**

Để minh hoạ cho bài toán thiết kế hệ vi xử lý chuyên dụng chúng ta sẽ thiết kế một hệ chuyên dụng có tên là Hệ thu tín hiệu ngẫu nhiên đa kênh. Chức năng của Hệ thu tín hiệu ngẫu nhiên đa kênh gắn liền với một yêu cầu

thực tế là tồn tại những hệ thống tự động phát hiện mục tiêu. Các mục tiêu là các đối tượng có các tham số không biết trước như cự ly, phương vị ...và sự xuất hiện của chúng. Các hệ hệ thống tự động phát hiện mục tiêu phải có khả năng bắt được thông tin (dạng bức xạ hay dạng phản xạ của tín hiệu mục tiêu) ở bất kỳ thời điểm nào. Một ví dụ sau đây cho phép ta hình dung một hệ thống như vậy.

Một hệ thống đặt ngầm ở dưới biển có chức năng phát hiện các mục tiêu theo các hướng phía trước, phía sau, bên phải, bên trái, phía trên và bên dưới dưới. Tín hiệu mục tiêu được coi là các chấn động thuỷ âm lan truyền tới hệ. Các chấn động này sẽ do các sensor cực nhạy phát hiện và tạo ở đầu ra là dạng xung điện áp (mức chuẩn TTL). Các xung này được dẫn vào các kênh tương ứng cho các hướng. Nhiệm vụ của hệ là bám sát và đếm tích luỹ số xung bắt được. Nếu số lượng đó đạt giá trị nào đó (trong ví dụ của chúng ta là 1 triệu xung) thì khẳng định là mục tiêu đã xuất hiện ở hướng tương ứng đó. Bản thân các tín hiệu này là các tín hiệu ngẫu nhiên cả về thời điểm xuất hiện cả về thời gian tồn tại của chúng. Hình 4.1. thể hiện sơ đồ hệ thống này.

4.2.2. Thiết kế hệ vi xử lý theo chức năng yêu cầu

Bước 1. Phân tích chức năng, nhiệm vụ hệ vi xử lý cần thiết kế

Nhiệm vụ của hệ vi xử lý cần thiết kế ở đây là hệ tự động phát hiện 6 kênh. Mỗi kênh hoạt động độc lập nhau. Tín hiệu trên mỗi kênh là dạng xung điện áp có các tham số là đại lương ngẫu nhiên. Căn cứ vào yêu cầu của nhiệm vụ như vậy, phần cứng phải được tổ chức trên hệ vi xử lý mới đủ khả năng thu thập và xử lý linh hoạt và mềm dẻo được. Đặc biệt là khối thu tin đa kênh phải được tổ chức sao cho có thể thu với sác xuất mất tin là thấp nhất và phải phản ứng kịp thời với tín hiệu ngẫu nhiên đi tới đầu thu. Phần mềm sẽ đảm nhiệm thuật toán điều khiển hoạt động của hệ theo chức năng như phương thức thu tin, phương thức xử lý tin...

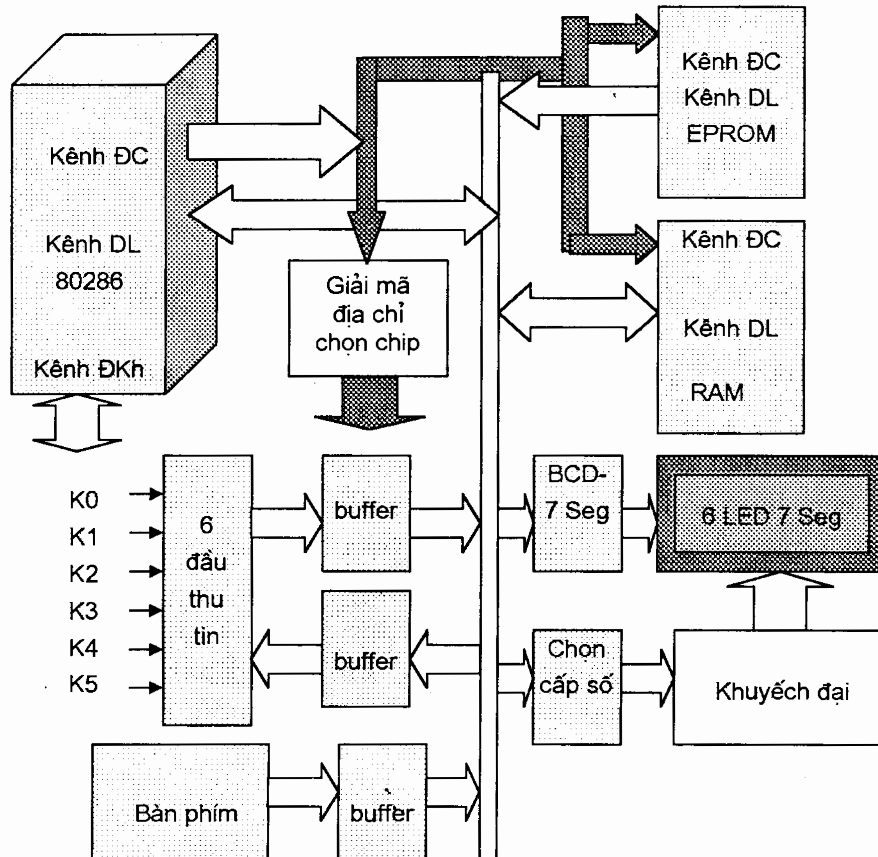
Bước 2. Tổ chức phần cứng cho hệ thu tín hiệu ngẫu nhiên 6 kênh.

Trên cơ sở của các phân tích của bước 1, phần cứng của Hệ thu tín hiệu ngẫu nhiên 6 kênh được tổ chức theo sơ đồ khối của hình 4.2, trong đó mỗi thành phần được thể hiện bằng một hộp chức năng. Các thành phần này liên hệ với nhau thông qua kênh hệ thống của hệ, đó là kênh địa chỉ, kênh dữ liệu và kênh điều khiển.

Bộ vi xử lý (CPU) được chọn là bộ vi xử lý 80286 vì nó có tốc độ đáp ứng yêu cầu cho hệ, có kênh dữ tiệu 16 bit nên quá trình xử lý sẽ thuận lợi cho cả cấu trúc byte và cấu trúc word. Khả năng quản lý không gian nhớ và quản lý các thiết bị ngoại vi của bộ vi xử lý 80286 hoàn toàn dủ cho cả việc mở rộng hệ thống.

Các thành phần bên trong gồm bộ nhớ trung tâm, bộ giải mã địa chỉ chọn chip và kênh dữ liệu. Bộ nhớ trung tâm gồm bộ nhứ ROM để chứa chương trình MONITOR và các tham số cố định như bảng địa chỉ cổng vào/ra, bảng mã tuyến tính... RAM để tạo bộ đệm dữ liệu cho các kênh và là nơi để chứa các kết quả trung gian trong quá trình thu thập, xử lý, gia công tín hiệu. RAM còn chứa các biến bộ nhớ của chương trình.

Bộ giải mã địa chỉ chọn chip có nhiệm vụ kích hoạt các thành phần có trong hệ mỗi khi bộ vi xử lý quy chiếu tới. Mỗi thành phần ít nhất phải có một tín hiệu *chip select* lấy từ bộ giải mã chọn chip.

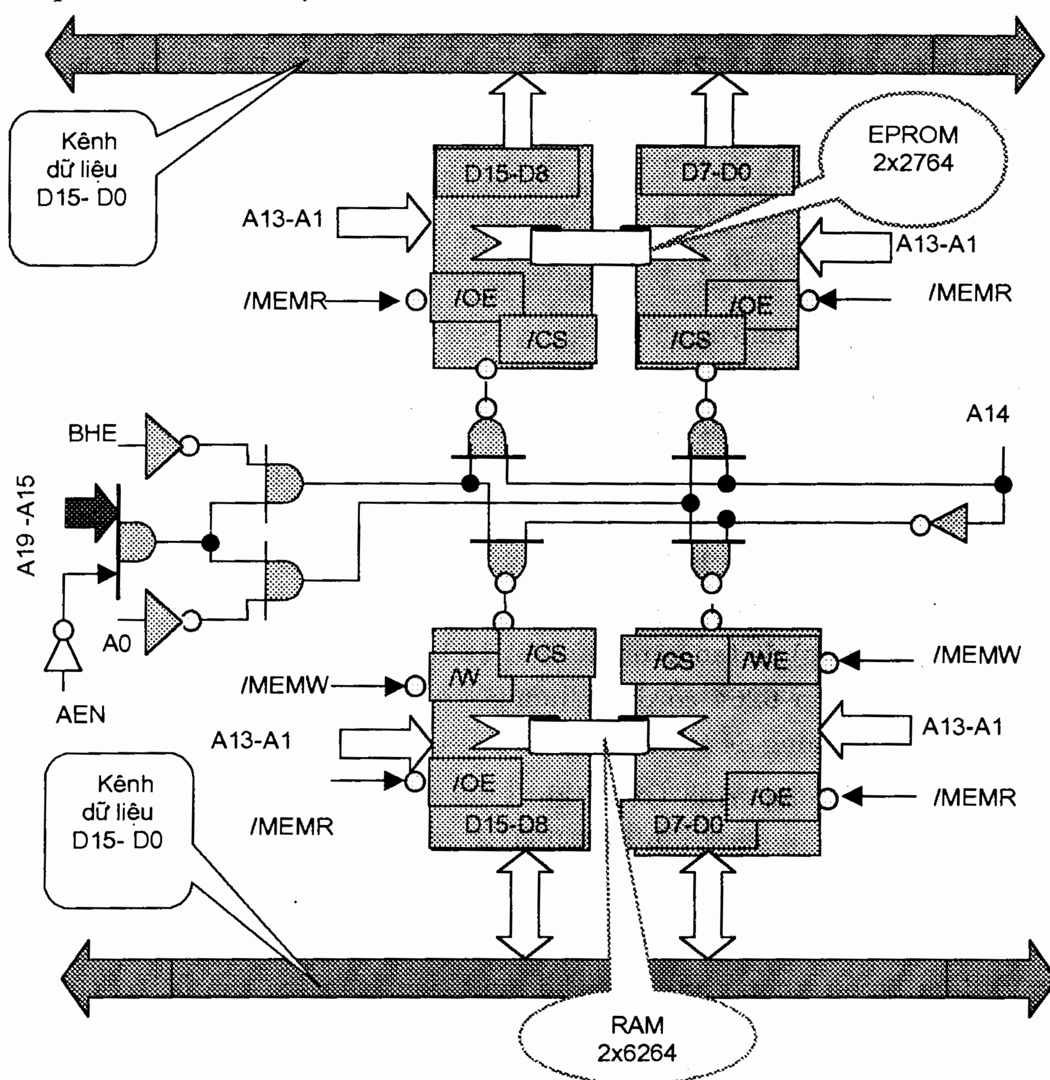


Hình 4.2. **Sơ đồ khối của hệ thu tin 6 kênh**

Các ngoại vi được tổ chức bao gồm ngoại vi thu tin 6 kênh để thu thập thông tin, ngoại vi hiển thị để hiển thị thông tin thu được của các kênh cho thao tác viên quan sát trực tiếp. Nó có dàn đèn hiển thị 6 cấp số thập phân để hiển thị giá trị cực đại theo điều kiện phát hiện mục tiêu là 1 triệu xung. Do khả năng của bộ vi xử lý nên chỉ cần dùng một bộ hiển thị chung cho cả 6 kênh. Việc ra lệnh cho kênh nào được hiển thị sẽ do ngoại vi bàn phím đảm nhiệm. Các thông tin của các ngoại vi này được bộ vi xử lý xử lý và phối hợp nhịp nhàng.

Kênh hệ thống gồm các kênh nội bộ và các kênh bên ngoài. Các bộ đệm kênh được tổ chức nhằm ngăn chặn các ảnh hưởng có thể có của ngoại vi tới kênh nội tại.

Để tổ chức kênh hệ thống nội tại cho hệ thu tín hiệu ngẫu nhiên 6 kênh dùng 80286 làm trung tâm điều khiển, chúng ta dựa vào sơ đồ nguyên lý thể hiện trên hình 2.8 có sử dụng các chip IC bổ trợ là bộ tạo giao động 82284 (hình 2.6), bộ điều khiển kênh (hình 2.7), mạch đệm địa chỉ 74AS533 (3 chip) và mạch đệm dữ liệu 74AS640 (2 chip). Với cách ghép nối như vậy, kênh hệ thống gồm kênh địa chỉ 24 bit, kênh dữ liệu 16 bit và kênh điều khiển với mọi tín hiệu giao tiếp cần thiết đã được hình thành.



Hình 4.3. **Tổ chức bộ nhớ trung tâm cho hệ vi xử lý 80286**

Tổ chức không gian nhớ

Vì địa chỉ khởi đầu của lệnh đầu tiên trong chế độ thực của CPU 80x86 là F000:FFF0 hex, nên ROM chứa chương trình Monitor phải nằm ở vùng nhớ cuối của MB cơ sở. Nếu ROM là 16Kb thì nó phải nằm từ địa chỉ F000:C000 đến F000:FFFF.

Còn RAM được phân bố tự do hơn, có thể bố trí ở bất kỳ vùng nào còn trống. Song, thường là bố trí chúng ngay dưới vùng ROM cho tiện quản lý (nếu không sử dụng chế độ ngắt).

Bảng 4.1

BIT:	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Đ/chỉ HEXA
	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F000 : 8000
	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	F000 : BFFF
	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	F000 : C000
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	F000 : FFFF

Do không gian nhớ thực của hệ vi xử lý cần thiết kế không chiếm hết không gian nhớ mà hệ 80286 có thể quản lý nên cần lựa chọn dung lượng phù hợp và sắp xếp hợp lý. Dung lượng cho ROM và cho RAM được chọn là 16 kB cho mỗi loại. Với dung lượng như thế hoàn toàn thoả mãn để chứa chương trình MONITOR và các vùng đệm dữ liệu cho chức năng hệ thống. Mặt khác, có thể dễ dàng chọn IC nhớ cho ROM như chip EPROM 2764 dung lượng 8 kB. Sử dụng 2 chip 2764 và mắc song song để tạo 2 băng nhớ (băng chứa byte cao và băng chứa byte thấp) để tạo kênh dữ liệu 16 bit. Cũng tương tự như vậy đối với RAM ta chọn 2 chip 6264 có dung lượng 8 kB cho mỗi chíp và kết hợp theo phương pháp song song để tạo 2 băng RAM.

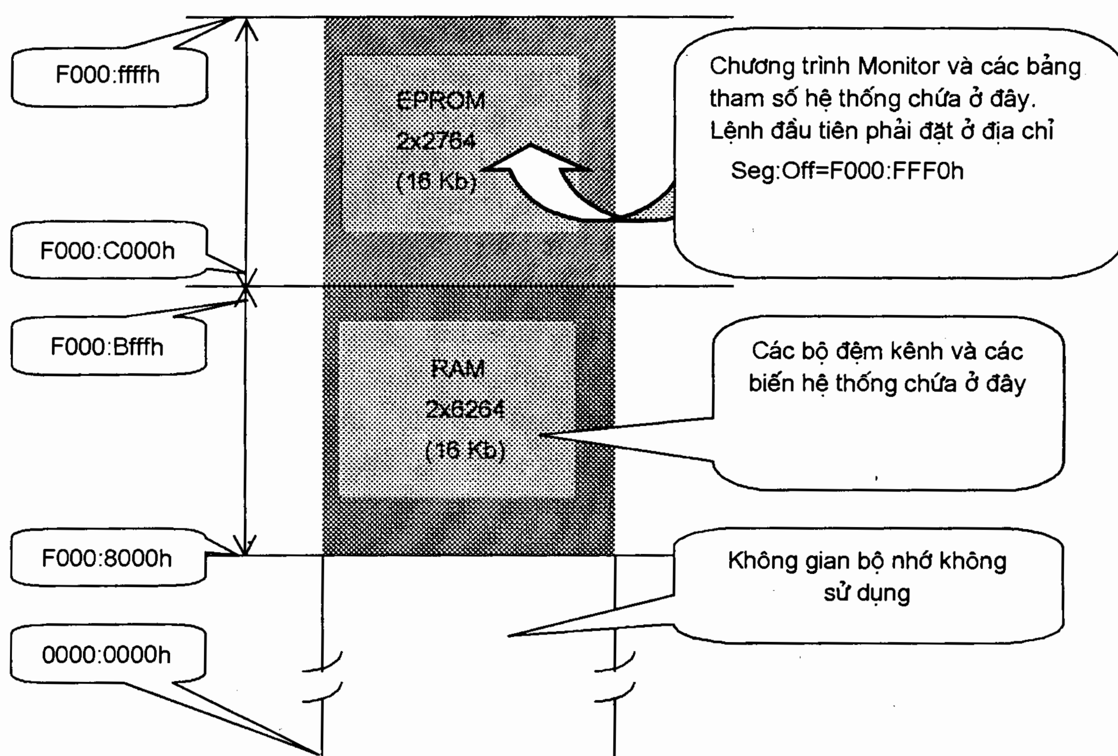
Vì con trỏ chương trình CS:IP khi khởi động có giá trị F000:FFF0, tức là lệnh đầu tiên của chương trình MONITOR phải chứa ở ngăn nhớ này. Chương trình MONITOR lại phải chứa trong ROM (EPROM) nên ROM phải chiếm không gian nhớ 16 kB của mảng 64 kB cuối cùng.

Vì điều khiển không dùng cơ chế ngắt nên RAM có thể bố trí tự do hơn. Song để tiện cho quản lý thì nên bố trí 16 kB RAM nằm sát ngay vùng ROM. Như vậy, cả vùng ROM và vùng RAM chiếm 32 kB của mảng 64 kB cuối cùng.

Bảng 4.1 là giá trị địa chỉ của vùng ROM và vùng RAM của hệ. Địa chỉ của vùng RAM bắt đầu từ giá trị F000: 8000H đến F000: BFFFH, còn địa chỉ của vùng ROM bắt đầu từ giá trị F000: C000H đến F000: FFFFH. Hình 4.3 là tổ chức bộ nhớ trung tâm cho hệ thu tín hiệu ngẫu nhiên 6 kênh.

Bảng 4.2

/BHE	A0	Chức năng điều khiển
0	0	Kênh dữ liệu 16 bit : D15 – D0
0	1	Kênh dữ liệu 8 bit cao : D15 – D8
1	0	Kênh dữ liệu 8 bit thấp: D7 – D0
1	1	Không sử dụng



Hình 4.4. **Tổ chức không gian nhớ của hệ vi xử lý**

Để quản lý được vùng nhớ vừa tổ chức cần tạo bộ giải mã địa chỉ cho phù hợp. Để con trỏ bộ nhớ luôn luôn trỏ tới nửa cuối của mảng nhớ cuối cùng thì các bít từ A19 đến A15 phải luôn bằng 1. Một mạch AND sẽ được sử dụng để tạo tín hiệu điều khiển này.

Để phân biệt vùng ROM và vùng RAM ta sử dụng bit A14. Khi A14 bằng 0 sẽ chọ vùng RAM, còn khi A14 bằng 1 sẽ chọn vùng ROM.

Để phân biệt băng chứa byte thấp và băng chứa byte cao ta sử dụng bit địa chỉ A0 và bit /BHE (Bus High Enable). Tổ hợp điều khiển này cho bởi bảng 4.2.

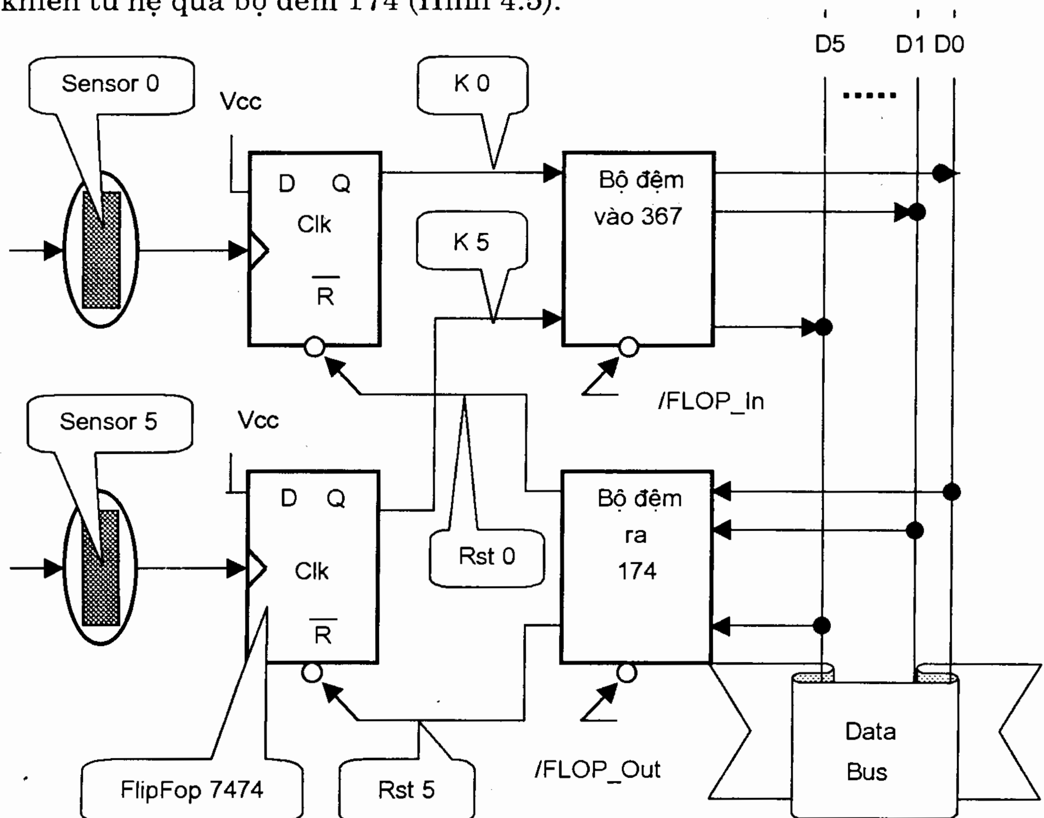
Các tín hiệu điều khiển từ CPU bao gồm AEN (Addr Enable), /MEMRD, /MEMWR phải tham gia vào bộ giải mã địa chỉ để tạo tín hiệu ra phù hợp.

Hình 4.4. là tổ chức không gian nhớ thực của hệ vi xử lý cần thiết kế. Như vậy, vùng nhớ rất lớn từ địa chỉ 0000:0000H đến F000:7FFFH không được sử

dụng. Nếu các thao tác hướng tới vùng này sẽ không được thực hiện và hệ có thể bị dừng hoạt động.

Tổ chức ngoại vi thu tin 6 kênh

Theo chức năng của ngoại vi thu tin thì có 6 kênh, mỗi kênh hoạt động độc lập. Kênh thứ nhất nối với sensor phía trước, kênh thứ hai nối với sensor phía sau, kênh thứ ba nối với sensor phía trên, kênh thứ tư nối với sensor phía dưới, kênh thứ năm nối với sensor phía phải và kênh thứ sáu nối với sensor phía trái. Dạng tín hiệu từ sensor đưa đến là tín hiệu ngẫu nhiên theo cả hai tham số là thời điểm xuất hiện và thời gian tồn tại nên mỗi đầu thu tin phải là dạng bẫy xung. Để tạo bẫy xung ta sử dụng mạch lật FlipFlop kiểu D để thực hiện chức năng này với cách tổ chức như sau: đầu vào D được gán logic 1 (nối với Vcc), đầu Clock nối với kênh vào. Như vậy, tại bất kỳ thời điểm nào, nếu xuất hiện một xung điện áp ở đầu vào thì FlipFlop sẽ lật ngay, đầu ra Q sẽ có logic 1 (theo chức năng của FlipFlop D). Tín hiệu này được đưa tới kênh dữ liệu hệ thống qua bộ đệm kênh 367. Lưu ý một điều là với cấu trúc này thì khi đã lật, FlipFlop D không tự trở về trạng thái ban đâù được nữa. Để lập lại trạng thái ban đầu cho FlipFlop D phải Reset cưỡng bức từ phía hệ vi xử lý. Đầu /R được nối với dây điều khiển từ hệ qua bộ đêm 174 (Hình 4.5).

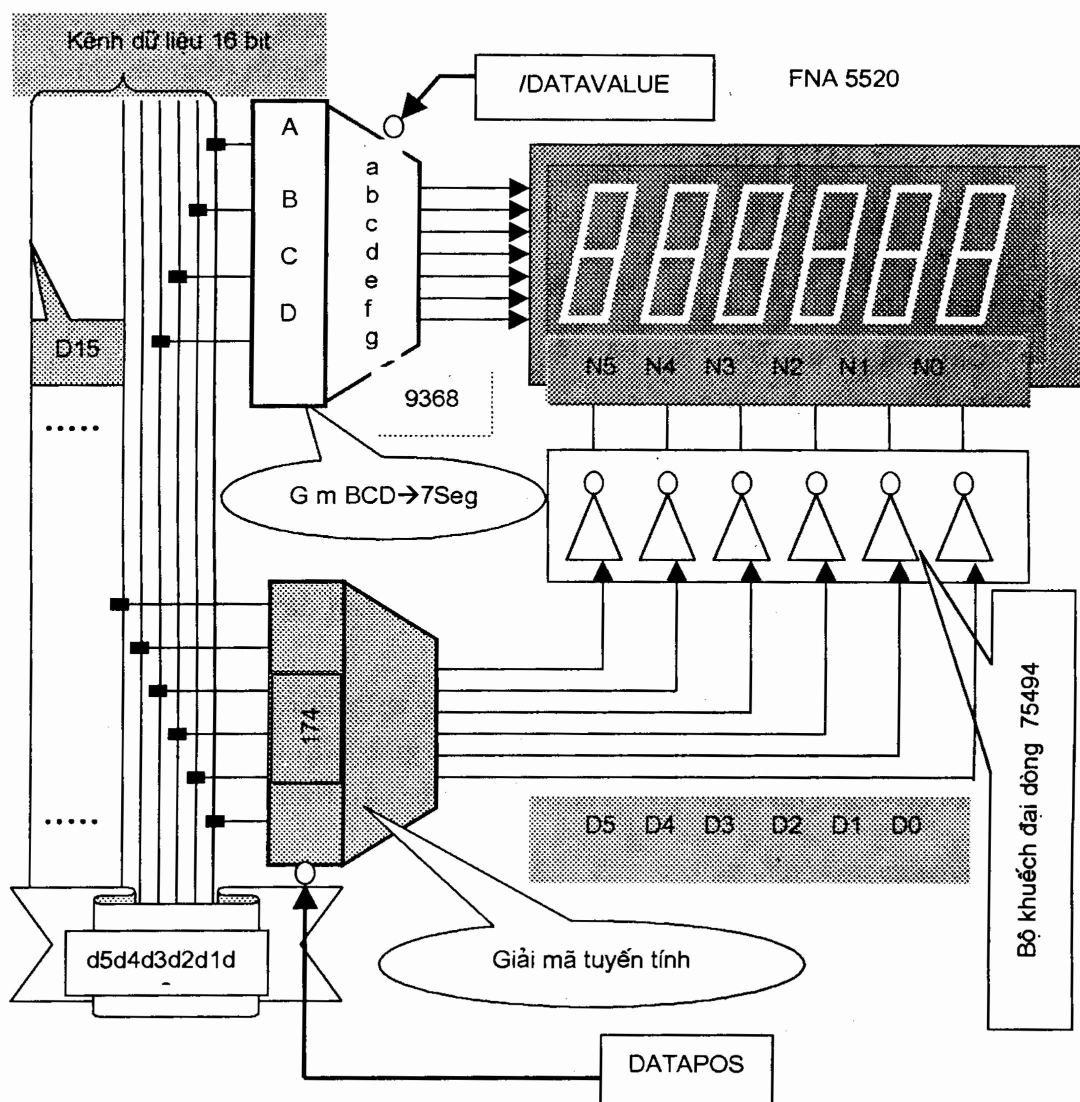


Hình 4.5. Tổ chức ngoại vi thu tin gồm 6 đầu thu

Tín hiệu vào/ra giữa ngoại vi với hệ vi xử lý được nối với 6 bit thấp của của kênh dữ liệu D0 đến D5. Như vậy khi xây dựng phần mềm cần nhớ là tin chỉ có trên 6 bit này còn các bit từ D6 đến D15 là không cần xét.

Các tín hiệu để điều khiển hoạt động của ngoại vi thu tin gồm tín hiệu Flop_In để điều khiển mở bộ đệm thu 367 để nhận tin từ các đầu thu, tín hiệu Flop_Out để điều khiển mở bộ đệm 174 để cấp từ mã lập trạng thái ban đầu cho các đầu thu (chỉ những đầu thu nào vừa nhận được tin). Các tín hiệu này phải lấy từ bộ giải mã địa chỉ chọn chip.

Tổ chức ngoại vi hiển thị thông tin kênh



Hình 4.6. **Tổ chức hệ hiển thị cho hệ vi xử lý**

Ngoại vi hiển thị thông tin kênh được tổ chức như hình 4.6 nhằm giúp thao tác viên nắm được số liệu của từng kênh khi hệ thống làm việc. Do khả năng về

tốc độ và quản lý của bộ vi xử lý nên chỉ cần tổ chức một dàn đèn hiển thị có 6 chữ số thập phân dùng chung cho cả sáu kênh. Dàn đèn 6 x7Seg được chọn là chip FNA 5520. Các đèn 7 Seg này mắc mạch theo dạng Katot chung. Để điều khiển việc hiển thị thông tin, hệ chỉ sử dụng một bộ giải mã BCD →7 Seg là chip 9368 dùng chung cho cả 7 đèn (là tổ hợp mạch chốt và mạch giải mã BCD→7 Seg).

Để chỉ định đèn nào được cấp thông tin, hệ sử dụng bộ giải mã tuyến tính 74174. Bộ mã tuyến tính là bộ mã đơn giản nhất và hệ phản ứng nhanh nhất với nó. Nguyên tắc cơ bản để sử dụng bộ mã này là số lượng các đối tượng cần điều khiển phải không lớn hơn số lượng các bit có trong kênh dữ liệu. Trong trường hợp của chúng ta, điều kiện này hoàn toàn thoả mãn (6 kênh thhông tin so với 16 bit kênh dữ liệu). Do vậy cấu trúc của bộ giải mã tuyến tính gồm chip chốt dữ liệu 6 bit 74174, chip khuyếch đại dòng 75494 nhằm đạt công suất hiển thị cho FNA 5520.

Như vậy muốn cho một đèn 7Seg của cấp số nào phát sáng thì trình tự điều khiển sau phải tuân thủ:

- Cấp giá trị mã BCD cho bộ giải mã BCD→7Seg.
- Kích hoạt chip 9368 để chuyển giá trị BCD trên 4 bit D3D2D1D0 sang mã 7seg ở đầu ra abcdefg bằng tín hiệu DataValue.
- Cấp giá trị mã tuyến tính cho bộ giải mã tuyến tính.
- Kích hoạt chip 74174 để chuyển giá trị mã tuyến tính trên 6 bit D5D4D3D2D1D0 sang đầu ra cho bộ khuyếch đại dòng 75494 bằng tín hiệu DataPos.

Vậy muốn cho cả 6 đèn 7Seg phát sáng thì điều khiển phải thực hiện quét vòng với tần số > 25 Hz nhằm đánh lừa thị giác. Khi tần số điều khiển quét đủ lớn thì ta có cảm giác là cả 6 đèn cùng lúc được cấp thông tin.

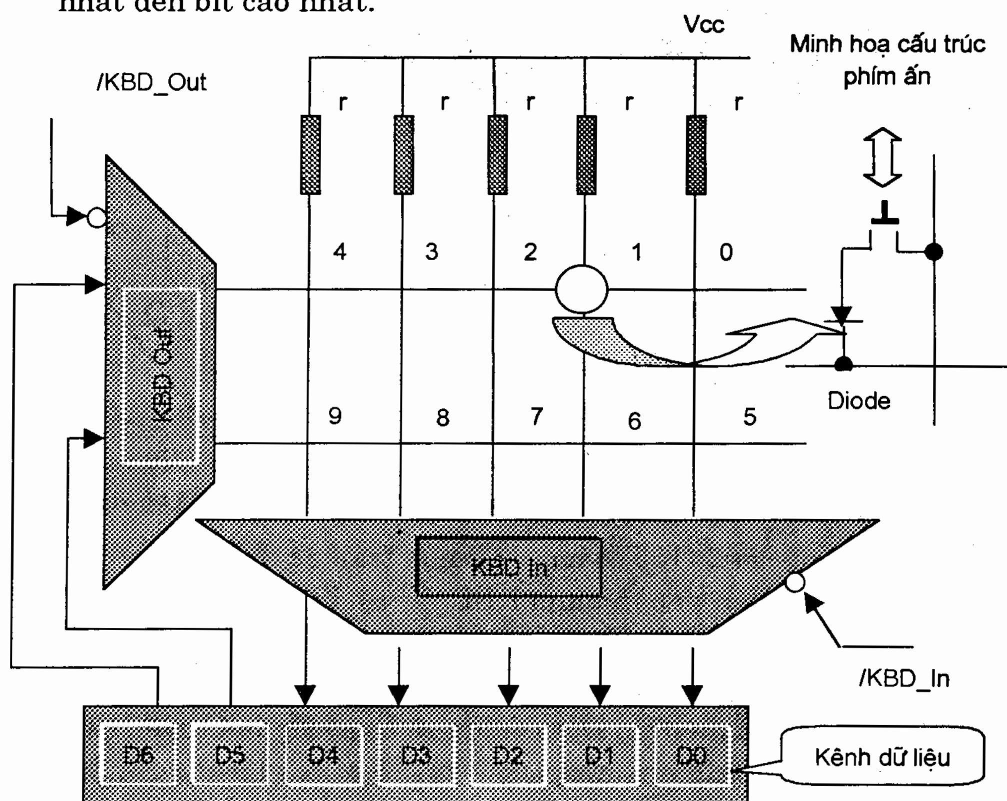
Tổ chức ngoại vi bàn phím

Bàn phím có chức năng đưa yêu cầu về kênh cần hiển thị giá trị lên dàn đèn hiển thị 6 cấp. Do vậy chỉ cần một bàn phím có số phím từ 6 trở lên là được. Một bàn phím như vậy được tổ chức như trên hình 4.7. Đó là ma trận 2x5 toạ độ, mỗi toạ độ là một phím. Mỗi phím gồm một công tắc mắc nối tiếp với một diode. Sơ đồ minh hoạ cấu trúc của phím số 2 (các phím khác có cấu trúc tương tự), ở trạng thái bình thường tiếp xúc bị hở mạch, diode lúc này bị thiên áp ngược nên đầu ra tương ứng bị đặt lên logic 1 do nó được nối với nguồn +Vcc. Trạng thái này là trạng thái thụ động. Khi phím bị ấn, tiếp xúc được gắn mạch, đầu ra tương ứng có giá trị logic 0 do dòng từ điện thế cao của nguồn +Vcc qua

điện trở hạn chế sẽ chảy qua diode lúc này đã được thiên áp thuận. Trạng thái này là trạng thái tích cực, trạng thái mà hệ hiểu là nó phải đưa giá trị của kênh có số hiệu bằng số hiệu của phím được ấn từ bộ đệm kênh lên dàn đèn hiển thị. Lưu ý là các tiếp điểm đã được xử lý chống rung cơ khí.

Như vậy muốn nhận dạng một phím được ấn thì trình tự điều khiển sau phải tuân thủ:

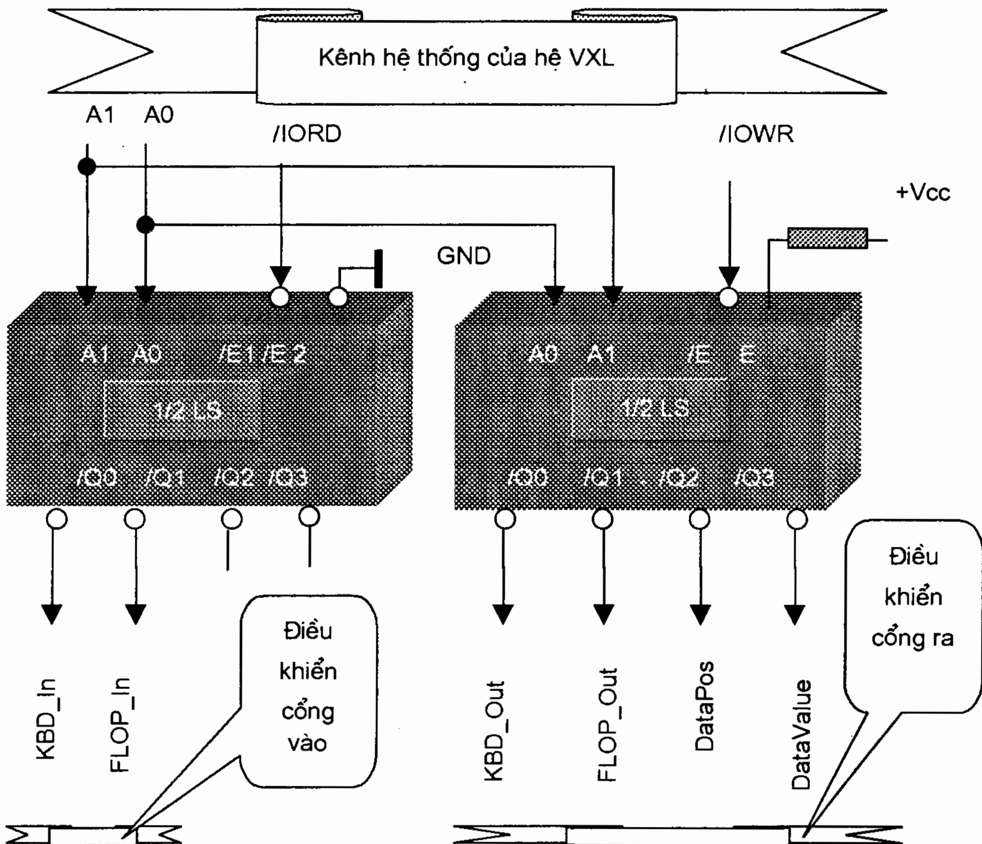
- Hệ vi xử lý phải cấp mã quét từ hàng thứ nhất đến hàng thứ hai của bàn phím. Mã quét này được chốt vào bộ đệm ra 74174 của bàn phím bằng tín hiệu KBD_Out. Đối với mỗi hàng, hệ vi xử lý phải đọc cả 5 phím vào thanh ghi AL bằng tín hiệu KBD_In của cổng vào.
- Ghép 5 phím hàng trên với 5 phím hàng dưới vào thanh ghi 16 bit nào đó.
- Duyệt từng bit để xem trạng thái phím được ấn. Trường hợp có hai phím trở lên bị ấn nhầm thì chỉ lấy phím có số hiệu thấp nhất.
- Đề phòng trường hợp bàn phím bị hỏng, để tránh cho chương trình bị quẩn hệ thống sẽ coi phím thứ sáu luôn luôn được ấn. Điều này không gây ảnh hưởng tới hoạt động thực của hệ vì phép duyệt lấy từ bit thấp nhất đến bit cao nhất.



Hình 4.7. **Tổ chức bàn phím cho hệ vi xử lý**

Tổ chức bộ giải mã chọn chip

Bộ giải mã chọn chip giúp bộ vi xử lý kích hoạt một cách đơn trị các thành phần có trong hệ. Mỗi thành phần có từ một địa chỉ trở lên. Nếu thành phần có nhiều địa chỉ thì mỗi địa chỉ sẽ quy chiếu một chức năng như chức năng chốt dữ liệu và chức năng mở cổng 3 trạng thái chẳng hạn. Tuy nhiên phần lớn các thành phần chỉ có một đầu chọn chip. Trong hệ thống này, bộ giải mã chọn chip dùng để điều khiển việc xuất/nhập thông tin cho các ngoại vi. Với các ngoại vi đã được xây dựng và phương thức điều khiển cho các ngoại vi đó thì phương án tổ chức bộ giải mã chọn chip tối ưu hơn cả là phương án như sơ đồ 4.8 đã chỉ ra.



Hình 4.8. **Tổ chức bộ giải mã cổng vào/ra để điều khiển ngoại vi của hệ vi xử lý**

Chip demultiplexer 74155 có hai nửa: nửa trái để điều khiển các cổng vào như cổng nhận điều khiển từ bàn phím KBD_In, nhận thông tin từ các kênh thu Flop_In, nửa phải để điều khiển các cổng ra như cổng cấp mã quét cho bàn phím KBD_Out, cổng lập trạng thái ban đầu cho các đầu thu Flop_Out, cổng cấp giá trị mã BCD cho dàn đèn hiển thị DataValue và cổng cấp mã tuyến tính cho dàn đèn hiển thị DataPos. Chip demultiplexer 74155 có hai bit địa chỉ lấy từ kênh địa chỉ A1A0. Tín hiệu điều khiển của bộ vi xử lý /IORD được nối với chân /E (Enable) của nửa trái để xác định hướng truyền tin vào còn tín hiệu điều khiển

của bộ vi xử lý /IOWR được nối với chân /E (Enable) của nửa phải để xác định hướng truyền tin ra.

Với tổ chức như vậy, bảng địa chỉ của từng cổng được thể hiện trên bảng 4.3.

Bảng 4.3

Tên cổng	Địa chỉ (Hex)	Hướng
KBD_In	00	Vào [In]
Flop_In	01	Vào [In]
KBD_Out	00	Ra [Out]
Flop_Out	01	Ra [Out]
DataPos	02	Ra [Out]
DataValue	03	Ra [Out]

Bước 3. Xây dựng phần mềm cho hệ thu tín hiệu ngẫu nhiên 6 kênh.

Như vậy phần cứng của hệ thu tín hiệu ngẫu nhiên đa kênh theo yêu cầu đã hình thành với hệ vi xử lý có bộ xử lý trung tâm là 80286 Intel và các ngoại vi cần thiết cho phép các chức năng thu nhận xử lý và hiển thị thông tin thu được. Trên cơ sở phần cứmg đó, phần mềm của hệ phải thực hiện các chức năng còn lại là :

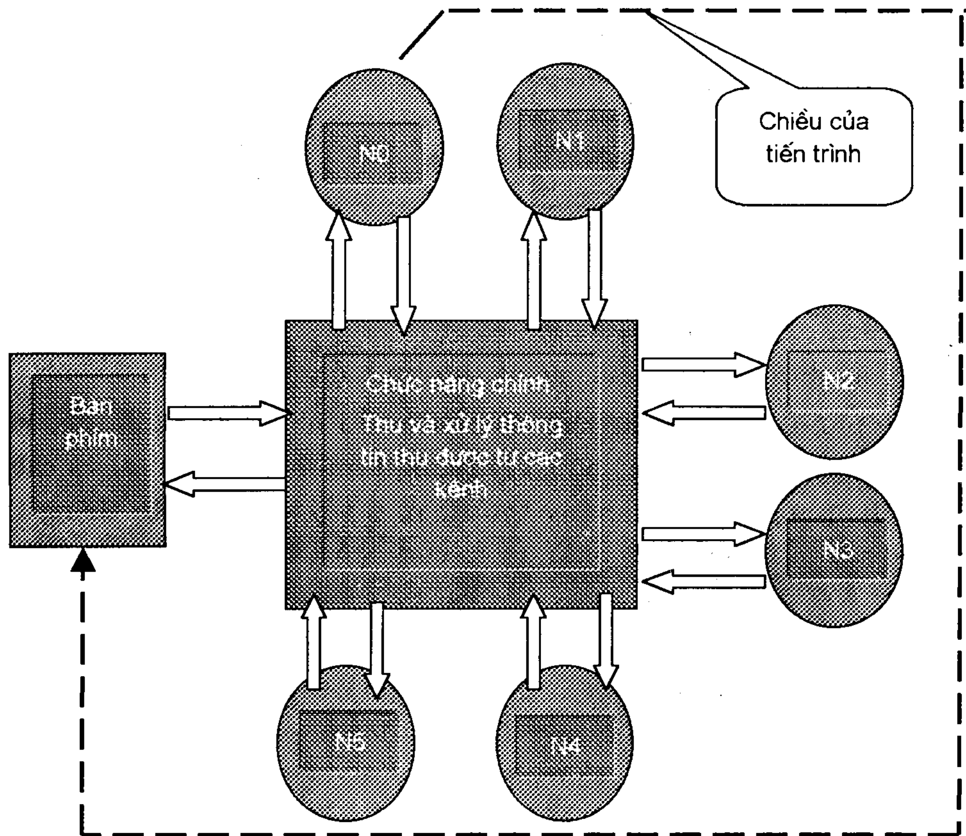
- Kiểm tra theo chu kỳ trạng thái các đầu thu nhằm xác định sự xuất hiện thông tin trên các kênh. Nếu ở kênh nào đó xuất hiện thông tin thì phải nhận dạng và chuyển thông tin đó vào vùng đệm tương ứng của kênh. Phương thức đưa thông tin vào vùng đệm tương ứng phải là dạng cộng tích luỹ nội dung cũ của vùng đệm với nội dung mới thu được, cụ thể:

  (MEMORY BUF) **mới** ← *(MEMORY BUF)* **cũ** +1.

  Kết quả lại phải để dưới dạng mã nhị thập phân BCD.
- Kiểm tra theo chu kỳ trạng thái của bàn phím nhằm xác định kênh cần hiển thị ở chu kỳ tiếp theo là kênh nào.
- Đưa nội dung của kênh cần hiển thị ra dàn đèn hiển thị. Lưu ý rằng bộ giải mã BCD→7Seg chỉ có một nên phương pháp quét phải được phần mềm thực hiện nhằm phân phối giá trị cho từng cấ số theo chu kỳ. Mặt khác phải thường xuyên kiểm soát các đầu thu cho nên thuật toán của phần mềm phải đặt mức ưu tiên cao nhất cho chức năng thu và xử lý tin thu được, thứ đến mới là chức năng hiển thị và chức năng kiểm soát trạng thái bàn phím. Thuật toán như vậy được thể hiện trên hình 4.9.

Căn cứ vào thuật toán thể hiện trên hình 4.9 rõ ràng là chức năng thu và xử lý tin được ưu tiên nhất. Khi chức năng hiển thị được thực hiện thì nó không hiển thị liên tục các cấp số mà mỗi lần hiển thị xong một cấp số nó lại phải trả

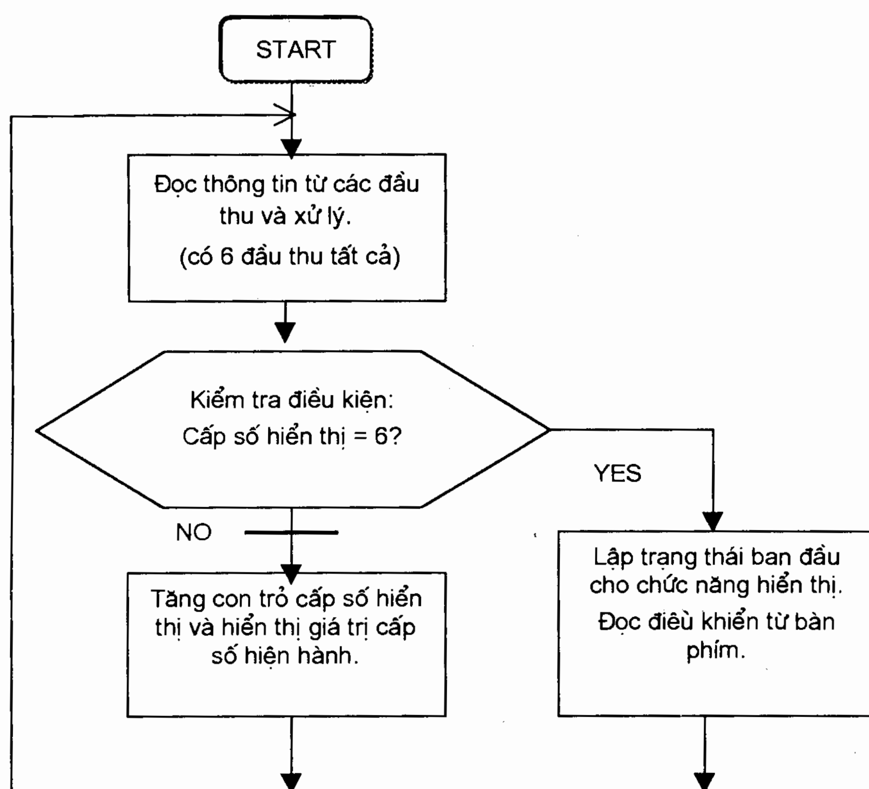
điều khiển về cho chức năng chính là chức năng thu và xử lý tin. Chỉ ở cuối chu kỳ hiển thị thì chức năng kiểm tra trạng thái bàn phím mới được thực hiện. Với thuật toán như vậy, hệ thống đảm bảo được một tiêu chuẩn quan trọng là sác xuất mất thông tin khi thu tin là thấp nhất.



Hình 4.9. **Thuật toán tổng quát của hệ thu tín hiệu ngẫu nhiên 6 kênh**

Trên cơ sở của thuật toán tổng quát của hệ thu tín hiệu ngẫu nhiên 6 kênh như vậy, lưu đồ thuật toán tổng quát cho lập trình được thể hiện trên hình 4.10. Với lưu đồ thuật toán tổng quát như thế có thể xây dựng chương trình MONITOR chính từ hai chương trình con là ***Thu_Xử lý*** để thu tin và xử lý tin cho các kênh và chương trình con ***HiểnThị_BF*** để hiển thị nội dung kênh và kiểm tra trạng thái bàn phím. Vòng lặp này sẽ xác định tần số cực đại cho các tín hiệu xung ở đầu thu tin. Như vậy bản thân chương trình MONITOR của hệ có thể viết:

```
CALL INIT ; các thao tác chuẩn bị cho hệ thống vào hoạt động
 LOOP_: CALL Thu_XL   ;gọi chương trình con Thu_Xử lý
        CALL HT_BF    ;gọi chương trình con Hiển Thị_BF
        JMP LOOP_     ;lặp lại từ nhãn LOOP_.
```



Hình 4.10. **Lưu đồ thuật toán tổng quát của hệ thu tín hiệu ngẫu nhiên 6 kênh**

Dữ liệu của từng kênh được lưu trữ trong bộ nhớ RAM. Trong RAM còn có các biến bộ nhớ như con trỏ cấp số Pointer và con trỏ kênh Channel. Tốt hơn cả là bố trí vùng đệm cho kênh trước còn sau là vùng dành cho biến con trỏ. Phương án tổ chức đó được thể hiện trên hình 4.11.

Mỗi kênh sử dụng 4 byte để chứa thông tin thu được. Ba byte đầu dùng để lưu trữ số thập phân có 6 chữ số, byte thứ tư dùng để chứa giá trị cờ CARRY của kênh. Nếu cờ này bằng 1 tức là một triệu đơn vị thông tin đã thu được. Đó cũng là dấu hiệu khẳng định mục tiêu đã xuất hiện ở hướng có kênh tương ứng.

Khởi đầu, vùng RAM này phải được dọn dẹp, tức gán giá trị 0 cho các ngăn nhớ. Thao tác này sẽ do phần INIT của chương trình chính đảm nhiệm. Phần INIT còn phải khởi tạo giá trị cho các con trỏ mảng tương ứng.

Tại phần đầu chương trình, cần khai báo các hằng:

```
RAM        EQU 8000H;     đ/chỉ đầu vùng RAM
POINTER    EQU  RAM+4*6 ;con trỏ cấp số
CHANNEL    EQU POINTER +1 ;con trỏ kênh
```

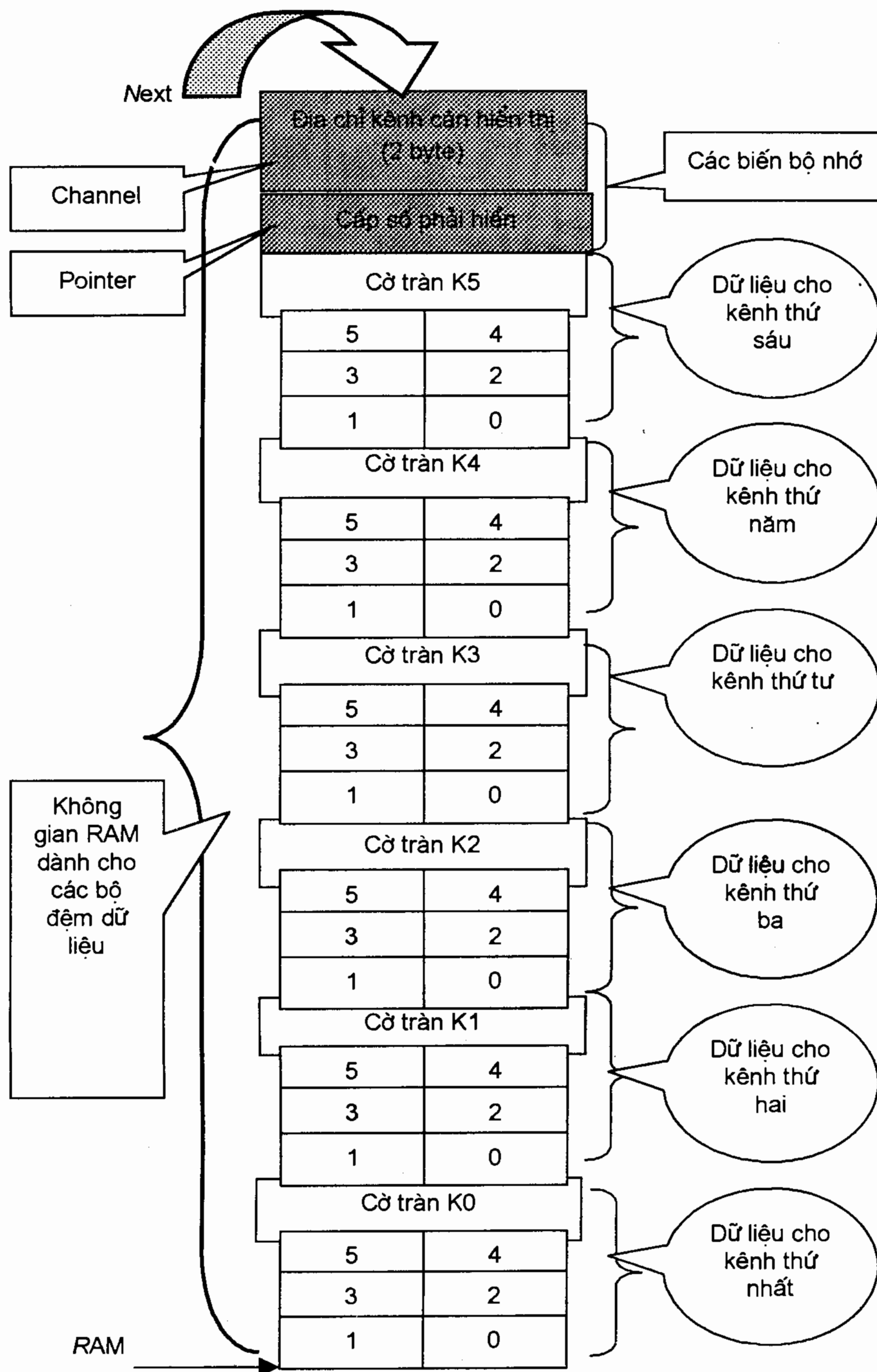
```
NEXT            CHANNEL+2; vùng tự do
EPROM       EQU 0C000H;     đ/chỉ đầu vùng ROM
TOP_STACK       EQU 0BFFFH;     đỉnh Stack
KBD     EQU  00H ; cổng bàn phím, đại diện cho cả
                 ; KBD_In và KBD_Out
FLOPS   EQU  01H  ; cổng thu tin và trạng thái thu tin, đại diện cho cả
                  ; Flop_In và Flop_Out
DataPos     EQU  02H    ; cổng điều khiển vị trí cấp số
DataValue EQU 03H      ;cổng đưa giá trị cần hiển thị
```

INIT của chương trình chính đảm nhiệm việc gán giá trị 0 cho các ngăn nhớ của RAM theo đoạn lệnh sau:

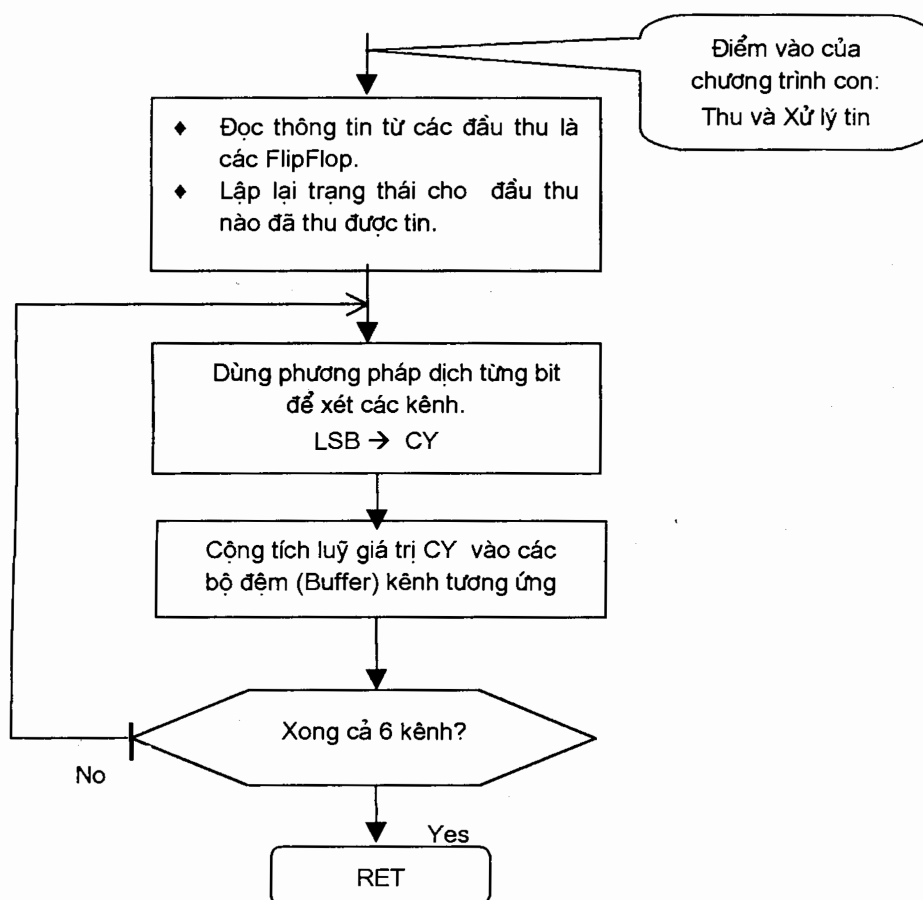
```
RESET: MOV CX, 6*4+1+2; 6*4 là dung lượng 6 kênh, 1 là dung lượng con
                        trỏ Pointer, 2 là dung lượng con trỏ Channel.
        MOV BX,0            ; con trỏ offset
LOOP_RAM:
        MOV [BX], 0         ; gán 0 cho ngăn nhớ
        INC BX              ; tăng con trỏ offset lên 1 đơn vị
        LOOP LOOP_RAM ; lặp lại
```

Lưu đồ thuật toán của chương trình con Thu_XL được thể hiện trên hình 4.12. Chương trình con thực hiện thuật toán này có nhiệm vụ kiểm tra một cách thường xuyên trạng thái của các đầu thu tin tức trạng thái của các FlipFlop. Đầu tiên, chương trình đọc thông tin vào thanh ghi AL, mỗi bít của AL sẽ chỉ rõ giá trị thu được của kênh tương ứng. Ngay sau khi nhận được thông tin, hệ phải lập lại trạng thái ban đầu cho đầu thu nào vừa thu được tin (lập có chọn lọc) để kịp bắt tín hiệu đến ngay sau đó.

Tiếp đó, hệ sẽ cộng tích luỹ giá trị mới nhận được vào nội dung cũ của vùng đệm kênh tương ứng. Mỗi khi xử lý xong một byte phải đổi thành dạng mã BCD. Chương trình sẽ xử lý lần lượt cả 6 kênh thì mới kết thúc chu kỳ kiểm soát thu tin.



Hình 4.11. **Tổ chức bộ đệm dữ liệu cho các kênh thông tin**



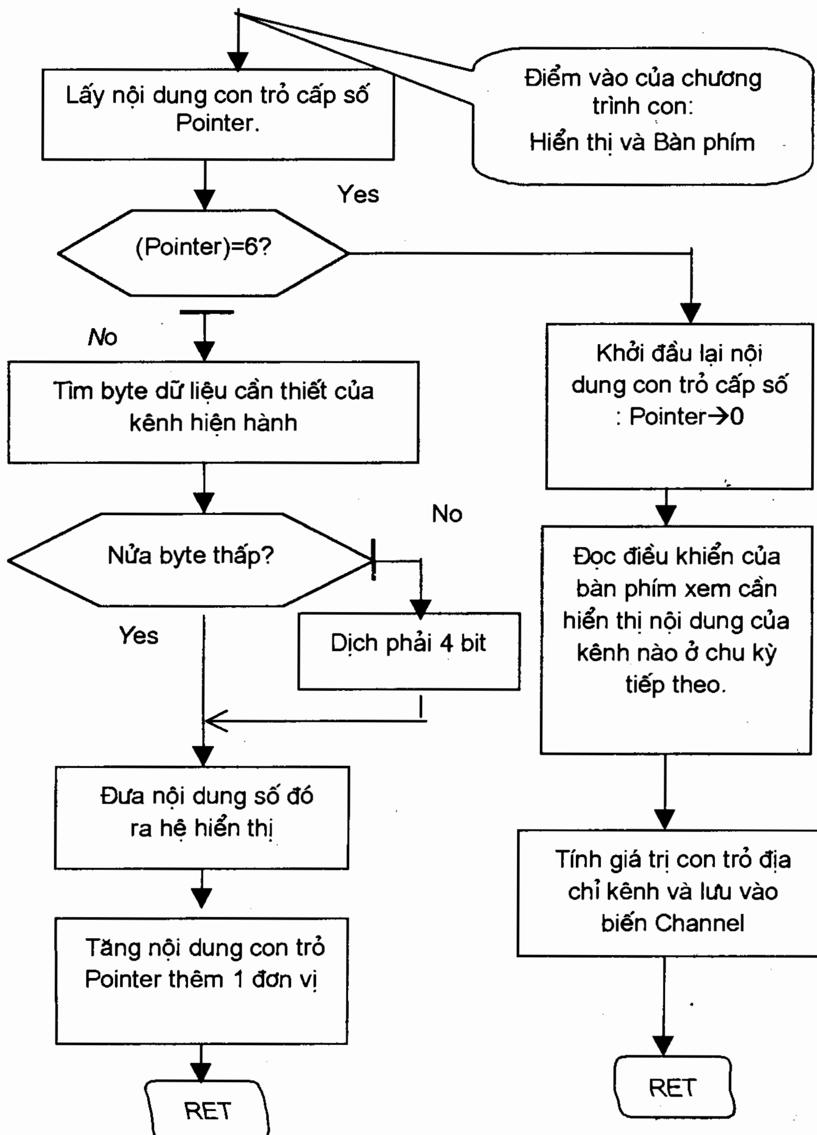
Hình 4.12. **Lưu đồ thuật toán chức năng Thu và Xử lý tin**

Chương trình nguồn của thuật toán có dạng:

```
;*********************************************
; Chương trình con này có nhiệm vụ:
;          -thu thông tin từ 6 kênh vào
;          -lập trạng thái cho các đầu thu
;          -xử lý (đổi mã BIN--->BCD), xắp xếp vào từng
;  vùng đệm cho kênh  tương ứng.
;*********************************************
Thu_XL PROC         ; mở thủ tục cho chương trình con
  PUSHA             ; cất giữ các thanh ghi
  IN AL,FLOPS       ; đọc nội dung các đầu thu vào AL
  MOV AH,AL         ;gửi tạm vào AH
  NOT AL            ; phủ định AL
  OUT FLOPS,AL      ; đưa ra các chân Clear cho các FlipFlop
```

```
 MOV AL,0FFH      ; byte FFh
 OUT FLOPS, AL    ; cho phép đầu thu hoạt động
 ;-----------------
 MOV CX,06H       ; bộ đếm 6
 MOV BX,RAM       ; BX là offset
LOOP_6K:          ; lặp cho 6 kênh
      MOV AL,AH        ; lấy lại giá trị cũ
      SHR AL,1         ;dịch phải 1 bit D0→CF
      MOV AH,AL        ; gửi tạm vào AH
      MOV AL,00H       ;gán AL=0
      ADC AL,[BX]      ; cộng có nhớ với nội dung ngăn nhớ đầu tiên
      DAA              ; chỉnh thập phân
      MOV [BX],AL      ; chuyển vào ngăn nhớ
      INC BX           ;tăng offet BX
      MOV AL,00H
      ADC AL,[BX]      ; cộng có nhớ với nội dung ngăn nhớ thứ hai
      DAA              ; chỉnh thập phân
      MOV [BX],AL      ; chuyển vào ngăn nhớ
      INC BX           ;tăng offet BX
      MOV AL, 00H
      ADC AL,[BX]      ; cộng có nhớ với nội dung ngăn nhớ thứ ba
      DAA              ; chỉnh thập phân
      MOV [BX],AL      ; chuyển vào ngăn nhớ
      INC BX           ;tăng offet BX trỏ tới thanh ghi cờ
      MOV AL,00H
      ADC AL,[BX]      ;đưa bit nhớ vào thanh ghi cờ
      MOV [BX],AL
      INC BX      ;tăng offet BX trỏ tới ngăn nhớ đầu của kênh tiếp theo
 LOOP LOOP_6K; lặp lại 6 lần
 POPA   ; khôi phục thanh ghi
 RET    ; kết thúc chương trình con, quay về trả quyền điều khiển cho
      ; chương trình chính
Thu_XL ENDP ; đóng thủ tục.
```

Lưu đồ thuật toán của chương trình con HT _BF được thể hiện trên hình 4.13. Chương trình con thực hiện thuật toán này có nhiệm vụ hiển thị nội dung kênh được ấn định bởi con trỏ kênh Channel. Mỗi khi chương trình con này thực hiện nó chỉ hiển thị một cấp số có số hiệu là nội dung của con trỏ Pointer. Nếu số hiệu cấp số đạt giá trị 6, nó sẽ chuyển điều khiển sang chức năng kiểm tra trạng thái của bàn phím. Trạng thái của bàn phím được đọc vào và tính toán thành địa chỉ của kênh cần hiển thị và chứa trong biến con trỏ Channel.



Hình 4.13. Lưu đồ thuật toán chức năng Hiển thị và Bàn phím

Chương trình nguồn của thuật toán có dạng:

```
;*********************************************
; Chương trình con này có nhiệm vụ:
;          -Hiển thi thông tin từ một  kênh được chỉ định
;   lên dàn đèn hiển thị theo phương pháp quét vòng.
;          -Đọc trạng thái bàn phím, tính địa chỉ cơ sở của kênh
;   cần hiển thị ở chu kỳ sau.
;*********************************************
HT_BF PROC        ;mở thủ tục
         PUSHA        ;lưu trữ thanh ghi
         MOV  AL,DS:[ POINTER]  ;chuyển byte offset=nội dung
                                ; Pointer vào AL
         CMP AL,06H          ;so sánh với 6
         JNC BF              ;đúng thì chuyển điều khiển sang bàn phím
         MOV AH,AL ;cất tạm vào AH
         AND AL,AL  ; xoá cờ CF
         SHR AL,1      ;dịch phải D0→CF
         PUSHF         ;cất cờ CF vào stack
         MOV  DX, Word Ptr DS:[ CHANNEL];chuyển 2 byte do
                                          ; CHANNEL trỏ tới vào DX
         ADD DL,AL     ;cộng chỉ số vào offset DX
         POPF          ;lấy lại cờ CF
         MOV  AL,[DX] ; chuyển byte đó vào AL
         JNC WAITE     ;nếu là chữ số chẵn thì chờ
         MOV CL,4      ; bộ đếm CL=4
         SHR AL,CL     ;dịch phải 4 bit  d7d6d5d4→d3d2d1d0
 OUTPUT:
         OUT DataVALUE,AL; đưa giá trị chữ số BCD ra BCD→7Seg
         MOV BX,offset TABLE  ;địa chỉ bảng mã tuyến tính
         ADD BL,AH           ;cộng với chỉ số kênh
         MOV  AL,[BX]        ;lấy byte do BX trỏ tới vào AL
         OUT DataPOS, AL  ;đưa ra bộ giải mã tuyến tính
         INC AH              ;tăng chỉ số cấp số cho vòng hiển thị sau
         MOV Byte Ptr DS:[POINTER], AH; chuyển vào biến Pointer
         POPA          ;khôi phục thanh ghi
         RET
```

```
WAITE: NOP              ; chờ
        JMP OUTPUT      ;nhảy về nhãn hiển thị
BF:
        MOV Byte Ptr DS:[ POINTER], 0 ;gán 0 cho biến Pointer
        MOV BX,RAM ;lấy địa chỉ của cả vùng đệm kênh
        MOV AL, 20h ;byte mã hàng thứ nhất cho bàn phím d6d5=01b
        OUT KBD,AL ;đưa ra ngoại vi bàn phím
        IN AL,KBD   ;nhận 5 phím hàng thứ nhất vào AL
        NOT AL
        MOV DL,AL ;chuyển sang DL
        MOV AL,40h ;byte mã hàng thứ hai cho bàn phím d6d5=10b
        OUT KBD,AL;đưa ra ngoại vi bàn phím
        IN AL,KBD   ;nhận 5 phím hàng thứ hai vào AL
        NOT AL
        MOV DH,AL ;chuyển sang DL
        AND DH,01h;chỉ giữ lại bit đầu là kênh thứ sáu
        MOV CL,5
        SHL DH,CL  ; dịch trái DH 5 bit
        OR DL,DH    ; bây giờ DL chứa trạng thái bàn phím cho sáu kênh
        OR DL,20H   ; mặc định kênh thứ sáu được hiển thị
        MOV CL, -1  ;bộ đếm tiến CL
LOOP_KBD:
        INC CL
        SHR DL,1
        JNC LOOP_KBD
        DONE:
        MOV AL,CL; CL đang chứa số hiệu kênh
        MOV AH,4
        MUL AH  ; nhân chỉ số DL với 4, kết quả trong AX
        ADD BX,AX  ;địa chỉ kênh cần hiển thị đã tính xong
        MOV Word Ptr DS:[CHANNEL],BX ;chuyển vào biến Channel
        POPA
        RET
```

```
TABLE:              ;bảng mã tuyến tính
        DB 01H
        DB 02H
        DB 04H
        DB 08H
        DB 10H
        DB 20H
 HT_BF ENDP         ;đóng thủ tục
;***************************************
```

Ghép các chương trình con lại ta được chương trình MONITOR cho hệ thu tín hiệu ngẫu nhiên đa kênh được viết bằng Assembly có dạng sau:

```
;------------Khai báo các thông số hệ thống -------------
        RAM         EQU 8000H ;     đ/chỉ đầu vùng RAM
        POINTER     EQU  RAM+4*6 ;con trỏ cấp số
        CHANNEL     EQU POINTER +1 ;con trỏ kênh
        NEXT        CHANNEL+2; vùng tự do
        EPROM       EQU 0C000H;     đ/chỉ đầu vùng ROM
        TOP_STACK           EQU 0BFFFH;     đỉnh Stack
        KBD     EQU  00H ; cổng bàn phím, đại diện cho cả
                    ; KBD_In và KBD_Out
        FLOPS  EQU  01H  ; cổng thu tin và trạng thái thu tin, đại
                    ;diện cho cả  Flop_In và Flop_Out
         DataPos   EQU  02H    ; cổng điều khiển vị trí cấp số
         DataValue EQU 03H   ;cổng đưa giá trị cần hiển thị
;------------------------------------------------------------------
.286                                ; sử dụng lệnh 80286
CODE_SEG SEGMENT  AT 0F000H ; mảng code_seg bắt đầu từ đây
        ASSUME CS:CODE_SEG
 ORG 100h
;--------------------------------
MAIN PROC
; ----khởi tạo: đặt các giá trị cho các thanh ghi quản lý mảng---
MOV AX,0F000H       ;địa chỉ mảng
```

```
        MOV DS,AX           ;chuyển vào DS
        MOV SS,AX           ;chuyển vào SS
        MOV ES,AX           ;chuyển vào ES
        MOV SP,TOP_STACK
        MOV BX,RAM
RESET: MOV CX, 6*4+1+2; 6*4 là dung lượng 6 kênh, 1 là dung
                    ;lượng con trỏ Pointer, 2 là dung lượng
                    ;con trỏ Channel.
        LOOP_RAM:
                MOV [BX], 0         ; gán 0 cho ngăn nhớ
                INC BX              ; tăng con trỏ offset lên 1 đơn vị
                LOOP LOOP_RAM ; lặp lại
 RECEIVER:
   CALL Thu_XL   ; gọi chương trình con thu và xử lý thông tin
   CALL HT _BF   ; gọi chương trình con hiển thị và điều khiển bàn phím
   JMP RECEIVER ; quay lại không điều kiện
MAIN ENDP
;***********************************************
; Chương trình con Thu_XL có nhiệm vụ:
;         -thu thông tin từ 6 kênh vào
;         -lập trạng thái cho các đầu thu
;         -xử lý (đổi mã BIN--->BCD), xắp xếp vào từng
;  vùng đệm cho kênh  tương ứng.
;***********************************************
Thu_XL PROC         ; mở thủ tục cho chương trình con
        PUSHA       ; cất giữ các thanh ghi
  IN AL,FLOPS       ; đọc nội dung các đầu thu vào AL
  MOV AH,AL         ;gửi tạm vào AH
  NOT AL            ; phủ định AL
  OUT FLOPS,AL      ; đưa ra các chân Clear cho các FlipFlop
  MOV AL,0FFH       ; byte FFh
  OUT FLOPS, AL     ; cho phép đầu thu hoạt động
 ;-----------------
```

```
    MOV CX,06H       ; bộ đếm 6
    MOV BX,RAM       ; BX là offset
  LOOP_6K:           ; lặp cho 6 kênh
        MOV AL,AH ; lấy lại giá trị cũ
        SHR AL,1     ;dịch phải 1 bit D0→CF
        MOV AH,AL ; gửi tạm vào AH
        MOV AL,00H ;gán AL=0
        ADC AL,[BX] ; cộng có nhớ với nội dung ngăn nhớ đầu tiên
        DAA             ; chỉnh thập phân
        MOV [BX],AL ; chuyển vào ngăn nhớ
        INC BX          ;tăng offet BX
        MOV AL,00H
        ADC AL,[BX]  ; cộng có nhớ với nội dung ngăn nhớ thứ hai
        DAA             ; chỉnh thập phân
        MOV [BX],AL  ; chuyển vào ngăn nhớ
        INC BX          ;tăng offet BX
        MOV AL, 00H
        ADC AL,[BX]   ; cộng có nhớ với nội dung ngăn nhớ thứ ba
        DAA              ; chỉnh thập phân
        MOV [BX],AL   ; chuyển vào ngăn nhớ
        INC BX           ;tăng offet BX trỏ tới thanh ghi cờ
        MOV AL,00H
        ADC AL,[BX]   ;đưa bit nhớ vào thanh ghi cờ
        MOV [BX],AL
        INC BX       ;tăng offet BX trỏ tới ngăn nhớ đầu của kênh tiếp theo
    LOOP LOOP_6K  ; lặp lại 6 lần
    POPA              ; khôi phục thanh ghi
    RET      ; kết thúc chương trình con, quay về trả quyền điều khiển cho
                 ; chương trình chính
  Thu_XL ENDP ; đóng thủ tục.
```

```
;*********************************************
; Chương trình con HT_BF có nhiệm vụ:
;           -Hiển thi thông tin từ một  kênh được chỉ định
;   lên dàn đèn hiển thị theo phương pháp quét vòng
;           -Đọc trạng thái bàn phím, tính địa chỉ cơ sở của kênh
;   cần hiển thị ở chu kỳ sau.
;*********************************************
HT_BF PROC        ;mở thủ tục
        PUSHA    ;lưu trữ thanh ghi
        MOV  AL,DS:[ POINTER];chuyển byte offset=nội dung Pointer
                     ;vào AL
        CMP AL,06H          ;so sánh với 6
        JNC BF        ;đúng thì chuyển điều khiển sang bàn phím
        MOV AH,AL ;cất tạm vào AH
        AND AL,AL  ; xoá cờ CF
        SHR AL,1     ;dịch phải D0→CF
        PUSHF        ;cất cờ CF vào stack
        MOV  DX, Word PTR DS:[ CHANNEL] ;chuyển 2 byte do
                                    ;CHANNEL trỏ tới vào DX
        ADD DL,AL  ;cộng chỉ số vào offset DX
        POPF          ;lấy lại cờ CF
        MOV  AL,[DX] ; chuyển byte đó vào AL
        JNC WAITE ;nếu là chữ số chẵn thì chờ
        MOV CL,4    ; bộ đếm CL=4
        SHR AL,CL  ;dịch phải 4 bit  d7d6d5d4→d3d2d1d0
 OUTPUT:
        OUT DataVALUE,AL; đưa giá trị chữ số BCD ra BCD→7Seg
        MOV BX,offset TABLE  ;địa chỉ bảng mã tuyến tính
        ADD BL,AH           ;cộng với chỉ số kênh
        MOV  AL,[BX]        ;lấy byte do BX trỏ tới vào AL
        OUT DataPOS, AL  ;đưa ra bộ giải mã tuyến tính
        INC AH              ;tăng chỉ số cấp số cho vòng hiển thị sau
        MOV Byte Ptr DS:[POINTER], AH  ; chuyển vào biến Pointer
        POPA          ;khôi phục thanh ghi
        RET
```

```
WAITE: NOP                ; chờ
        JMP OUTPUT        ;nhảy về nhãn hiển thị
BF:
        MOV Byte Ptr DS:[ POINTER], 0 ;gán 0 cho biến Pointer
        MOV BX,RAM        ;lấy địa chỉ của cả vùng đệm kênh
        MOV AL,20h ;byte mã hàng thứ nhất cho bàn phím d6d5=01b
        OUT KBD,AL        ;đưa ra ngoại vi bàn phím
        IN AL,KBD         ;nhận 5 phím hàng thứ nhất vào AL
        NOT AL
        MOV DL,AL ;chuyển sang DL
        MOV AL,40h ;byte mã hàng thứ hai cho bàn phím d6d5=10b
        OUT KBD,AL        ;đưa ra ngoại vi bàn phím
        IN AL,KBD  ;nhận 5 phím hàng thứ hai vào AL
        NOT AL
        MOV DH,AL         ;chuyển sang DL
        AND DH,01h;chỉ giữ lại bit đầu là kênh thứ sáu
        MOV CL,5
        SHL DH,CL; dịch trái 5 bit
        OR DL,DH   ; DL bây giờ là trạng thái bàn phím cho sáu kênh
        OR DL,20H  ; mặc định kKênh thứ sáu được hiển thị
        MOV CL, -1  ;bộ đếm tiến CL
LOOP_KBD:
        INC CL
        SHR DL,1
        JNC LOOP_KBD
DONE:
        MOV AL,CL        ; CL đang chứa số hiệu kênh
        MOV AH,4
        MUL AH ; nhân chỉ số DL với 4, kết quả trong AX
        ADD BX,AX ;địa chỉ kênh cần hiển thị đã tính xong
        MOV Word Ptr DS:[CHANNEL], BX ;chuyển vào biến Channel
        POPA
        RET
```

```
TABLE:              ;bảng mã tuyến tính
          DB 01H  ; cho đèn cấp số N0
          DB 02H
          DB 04H
          DB 08H
          DB 10H
          DB 20H
HT_BF ENDP            ;đóng thủ tục
;-----------------------------------------------
CODE_SEG  ENDS
END MAIN
```

Bước 4. Nạp chương trình cho hệ thu tín hiệu ngẫu nhiên đa kênh.

Trên cơ sở của các kết quả của bước 1, bước 2 và bước 3, ở bước 4 phải nạp chương trình cho hệ vi xử lý cần thiết kế và hiệu chỉnh chức năng của nó.

Tiến hành mô phỏng chương trình đã xây dựng ở bước 3 trên hệ vi xử lý phát triển. Khi đó, mọi trạng thái của quá trình hoạt động của hệ vi xử lý được ghi nhận và thông báo. Người thiết kế căn cứ vào đó để hiệu chỉnh cả phần cứng và phần mềm cho thật chính xác.

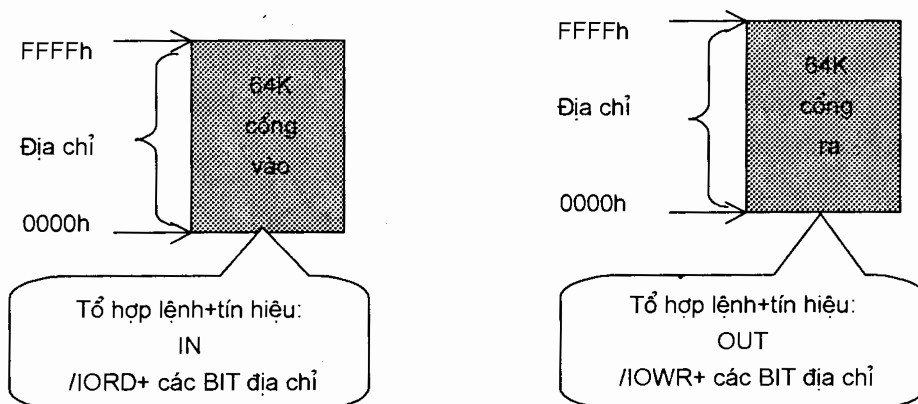
Sử dụng máy nạp ROM để nạp chương trình đã biên dịch vào ROM hay EPROM của hệ vi xử lý và cho chạy. Quá trình chạy thử người thiết kế phải theo dõi để tiến hành điều chỉnh cho tới khi hệ hoạt động hoàn toàn thoả mãn các yêu cầu đặt ra.

Chương 5
CỔNG TRAO ĐỔI THÔNG TIN VỚI NGOẠI VI

Các hệ vi xử lý luôn luôn đòi hỏi sự trao đổi dữ liệu với các ngoại vi. Trong các chương trước đã đề cập phần nào tới cổng trao đổi thông tin giữa hệ vi xử lý với các thiết bị ngoại vi. Các thông số cơ bản khi tổ chức một cổng là địa chỉ của cổng, hướng truyền thông tin của cổng và độ rộng kênh dữ liệu của cổng. Trong chương này sẽ đi sâu vào nghiên cứu và tổ chức cổng trao đổi dữ liệu. Có nhiều phương pháp điều khiển sự trao đổi dữ liệu này như vào/ra điều khiển bằng chương trình, vào/ra điều khiển bằng ngắt, vào/ra điều khiển bằng phần cứng. Trong nhóm thứ nhất, chương trình sẽ nắm quyền điều khiển quá trình trao đổi dữ liệu vào/ra. Nhóm thứ hai là nhóm ngoại vi chủ động đưa yêu cầu cho hệ vi xử lý khi ngoại vi có nhu cầu trao đổi dữ liệu. Nhóm thứ ba là nhóm trao đổi dữ liệu được điều khiển bằng phần cứng.

5.1. VÀO/RA THÔNG TIN TÁCH BIỆT

Chỉ có lệnh IN và lệnh OUT thực hiện trao đổi dữ liệu trong chế độ vào/ra thông tin tách biệt. Bộ vi xử lý 80x86 sử dụng 16 bit địa chỉ để gán địa chỉ cho các cổng vào/ra nên nó quản lý được 64 K cổng vào và 64K cổng ra. Với số lượng như vậy, thì thực tế có thể coi là hoàn toàn đáp ứng mọi yêu cầu về giao tiếp với ngoại vi. Thực tế rất ít khi sử dụng tới giá trị cực đại về số lượng đó. Hình 5.1 thể hiện cơ chế đánh địa chỉ cho các cổng vào/ra.



Hình 5.1. **Tổ chức cổng vào/ra của 80x86**

Lệnh IN và OUT đòi hỏi một số chu kỳ máy để thực hiện: chu kỳ máy thứ nhất là chu kỳ nhận lệnh OF, chu kỳ máy thứ hai để đưa địa chỉ cổng từ thanh ghi DX (nếu địa chỉ này >255) hay địa chỉ trực tiếp ghi trong lệnh ra kênh địa chỉ. Chu kỳ máy là IORD hoặc IOWR. Trong chu kỳ máy này, lệnh OUT gửi dữ liệu từ bộ vi xử lý ra thiết bị ngoại vi, lệnh IN nhận dữ liệu từ thiết bị ngoại vi vào bộ vi xử lý. Trong chu kỳ máy IORD hoặc IOWR, các tín hiệu /IORD và /IOWR xác định thời điểm để chốt cổng ba trạng thái hoặc để chốt cổng ra.

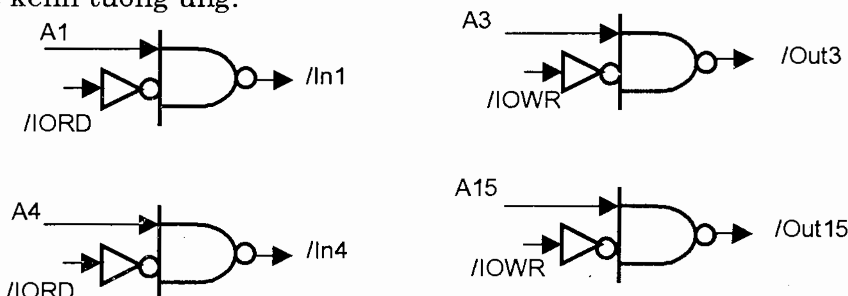
Đối với các cổng vào, bộ giải mã chọn chip có đầu vào là các tín hiệu /IORD và địa chỉ cổng sẽ tạo ra xung chọn thiết bị vào duy nhất cho mỗi cổng vào. Xung này chỉ xuất hiện trong chu kỳ máy IORD của lệnh IN. Xung chọn thiết bị vào mở các bộ đệm ba trạng thái của cổng vào (hình 5.2).

Đối với các cổng ra, bộ giải mã có các đầu vào là /IOWR và địa chỉ cổng ra sẽ tạo ra xung chọn thiết bị ra để chọn cổng ra. Xung này chỉ xuất hiện trong chu kỳ IOWR của lệnh OUT. Thông thường, mỗi xung chọn thiết bị ra làm nhịp cho một cổng ra. Có thể có nhiều cổng ra được chọn đồng thời.

Sơ đồ phần cứng chọn thiết bị được thiết kế phụ thuộc vào số lượng thiết bị vào/ra mà hệ yêu cầu. Không cần phải giải mã địa chỉ cổng nếu hệ chỉ có một cổng vào và một cổng ra. Tín hiệu điều khiển với /IORD sẽ tạo xung đọc và với /IOWR để tạo xung ghi. Các xung nhịp này điều khiển trực tiếp thanh ghi đệm vào và thanh ghi chốt ra. Nếu trong hệ có nhiều cổng vào và cổng ra, cần phải giải mã địa chỉ để tạo xung chọn thiết bị cho mỗi cổng.

Phương pháp giải mã địa chỉ đơn giản nhất là chọn tuyến tính. Phương pháp này không đòi hỏi phần cứng, song chỉ sử dụng cho hệ có tổng số cổng vào/ra không lớn hơn 16. Ở đây mỗi bit địa chỉ có quan hệ loại trừ với cổng vào/ra và tổ hợp với giá trị lôgíc /IORD hoặc /IOWR để tạo ra xung chọn thiết bị.

Chú ý rằng xung chọn thiết bị trong chu kỳ máy ở thời điểm của xung /IORD hoặc /IOWR xuất hiện trừ đi độ trễ của mạch lôgíc chọn. Trên hình 5.2, bit địa chỉ A4=1 thì cổng IN16 sẽ được chọn. Xung chọn thiết bị vào /Out16 được sinh ra từ bit A4 và /IORD. Đầu ra /Out16 cho phép bộ đệm ba trạng thái kích hoạt kênh tương ứng.



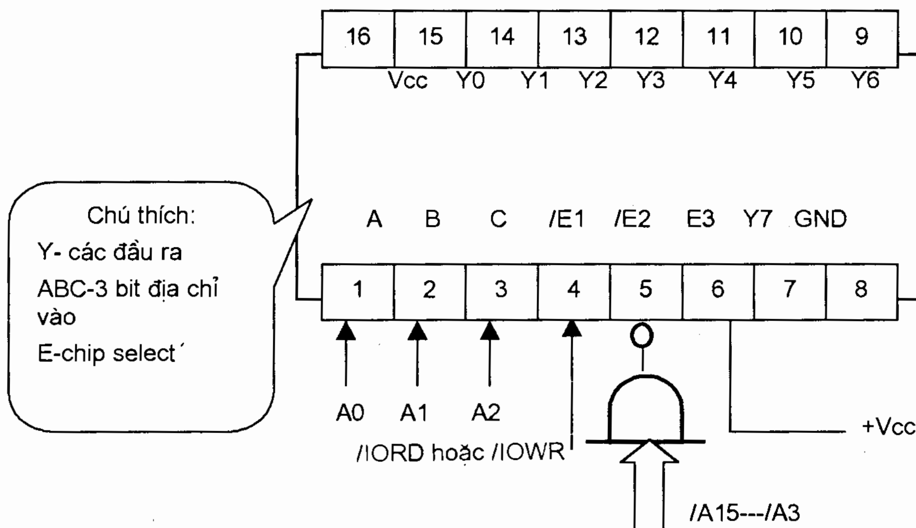
Hình 5.2. **Tạo tín hiệu chọn chip bằng giải mã tuyến tính**

Bất lợi của phương pháp chọn tuyến tính là lỗi chương trình sẽ gây nguy hiểm cho phần cứng. Nhầm lẫn địa chỉ cổng sẽ gây ra hai hay nhiều cổng vào đồng thời kích hoạt kênh dữ liệu.

Phương pháp chọn tuyến tính giới hạn tổng số cổng vào/ra của hệ. Để chọn được nhiều cổng vào/ra hơn, cần có bộ giải mã địa chỉ. Tổng số cổng dành cho thiết bị ngoại vi là 65536 cho lệnh *IN* và 65536 cho lệnh *OUT*.

Khi giải mã các bit địa chỉ, để có nhiều xung chọn thiết bị vào, ta cần các bộ giải mã có nhiều đầu vào bởi vì như vậy sẽ giảm số lượng IC cần thiết. Nếu một bộ yêu cầu bốn cổng vào và bốn cổng ra, có thể sử dụng IC 74LS139. IC74LS139 có hai bộ giải mã, mỗi bộ có bốn đầu ra.

Các bộ giải mã có sáu đầu vào có chức năng rộng lớn hơn như các bộ 8205 và 74LS138 (hình 5.3). Khi bộ 8205 hoặc 74LS138 được sử dụng để phát sinh xung chọn thiết bị cho hệ vi xử lý, các tín hiệu chốt /IORD hoặc /IOWR phải nối với một trong các đầu vào "chọn mạch" với mức tích cực âm, còn đầu vào "chọn mạch" với mức tích cực âm kia được nối với tổ hợp các bit địa chỉ cao A15-A3. Tín hiệu E được nối với +Vcc để có mức tích cực dương.



Hình 5.3. **Bộ giải mã chọn chip dùng Demux 74138: Chọn cổng vào thì sử dụng /IORD còn chọn cổng ra thì sử dụng /IOWR**

Các bộ giải mã có đầu vào "chọn mạch" mắc nối tiếp nhau cho phép tạo nhiều hơn các tín hiệu "chọn thiết bị". Hình 5.4 là sơ đồ tạo 256 tín hiệu chọn thiết bị từ các bộ giải mã 16 đầu ra 74LS154.

Hình 5.4. **Bộ giải mã chọn chip dùng Demux 741LS154 tạo 256 cổng vào**

5.2. VÀO/RA THÔNG TIN THEO ĐỊA CHỈ BỘ NHỚ

Các lệnh qui chiếu bộ nhớ cũng có khả năng trao đổi dữ liệu giữa thiết bị ngoại vi với bộ vi xử lý nếu cổng vào/ra được gán cho vùng địa chỉ của bộ nhớ. Các thanh ghi liên quan tới cổng vào/ra được coi như là các ngăn nhớ.

Nếu chọn dây địa chỉ A15 để phân biệt quá trình quy chiếu bộ nhớ và quá trình quy chiếu cổng vào/ra thì A15 =0 nghĩa là các ngăn nhớ được địa chỉ hoá, còn A15 = 1, các thanh ghi cổng vào/ra theo địa chỉ bộ nhớ được địa chỉ hoá. Như vậy có 32K byte của vùng địa chỉ bộ nhớ dành cho bộ nhớ và 32 Kbyte còn lại được dành cho vào/ra theo địa chỉ bộ nhớ. Mạch logic bên ngoài sinh ra xung chọn thiết bị vào/ra chỉ khi các tín hiệu điều khiển có các giá trị tương ứng /MEMRD = 0 hoặc /MEMWR = 0.

Các lệnh trao đổi nội dung với bộ nhớ của bộ vi xử lý 80x86 đều được dùng trong trao đổi nội dung với các cổng theo địa chỉ bộ nhớ.

Khi bộ vi xử lý gửi địa chỉ và các tín hiệu điều khiển để đọc thông tin từ bộ nhớ, nó không cần xác định nơi gửi dữ liệu vào là bộ nhớ hay thiết bị vào/ra. Bộ vi xử lý chỉ cần nơi gửi dữ liệu duy trì khoảng thời gian tối thiểu để dữ liệu có

thể ổn định hoặc nó sử dụng đầu vào READY để tạo thêm một số trạng thái WAIT cần thiết. Tương tự như vậy, khi bộ vi xử lý ghi thông tin vào bộ nhớ, nó chỉ cung cấp địa chỉ, dữ liệu và xung đồng bộ rồi tiếp tục chức năng của nó. Mạch lôgíc bên ngoài sẽ xác định là bộ nhớ hay cổng vào/ra nào phải tiếp nhận dữ liệu đưa ra.

Các cổng vào/ra có thể là các vi mạch cỡ nhỏ, cỡ vừa hoặc cỡ lớn (MSI và LSI). Một số vi mạch MSI là các cổng vào/ra trên một tấm đơn tinh thể. Một cổng ra thông thường là thanh ghi đơn giản hoặc thanh ghi chốt dữ liệu. Hình 5.5 biểu diễn sơ đồ của hai thanh ghi chốt 8 bit 74LS373 và 74LS374.

Thanh chốt 74LS373 gồm 8 chốt theo ***mức dương*** loại D có một đầu vào mở mạch. Để kích đầu mở mạch, cần có xung chọn ở mức lôgíc 1. Xung này được sinh ra bởi giá trị của xung chọn thiết bị ra mức lôgíc 0 hoặc được sinh ra một cách trực tiếp.

20	19	18	17	16	15	14	13	12	11
Vcc	8Q	8D	7D	7Q	6Q	6D	5D	5Q	CE
/OE	1Q	1D	2D	2Q	3Q	3D	4D	4Q	GND
1	2	3	4	5	6	7	8	9	10

Chức năng	CE	/OE	D	Q
Đệm	H	L	H	H
Đệm	H	L	L	L
Chốt	L	L	X	Q
Trở kháng cao	X	H	X	Hi-z

***Hình 5.5.* Sơ đồ chân và bảng trạng thái của 74LS373 (374)**

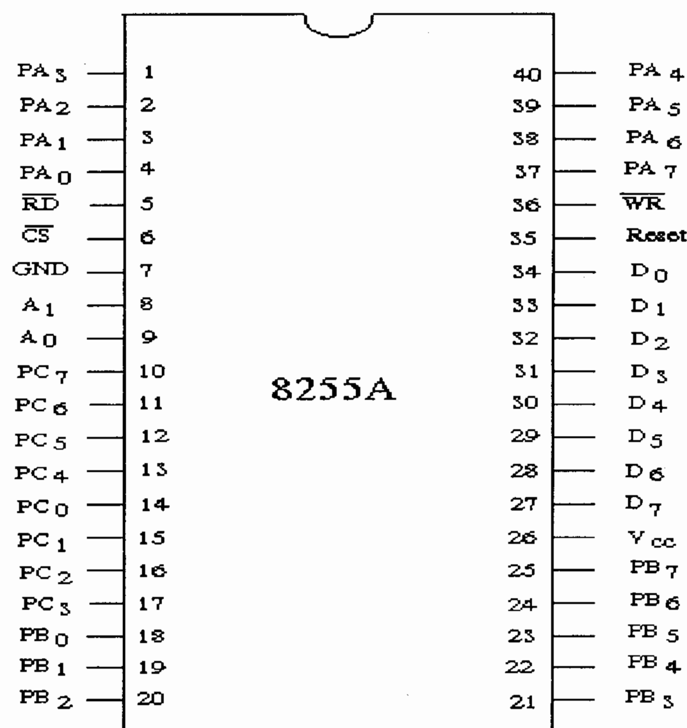
Thanh chốt 74LS374 gồm 8 chốt theo ***sườn dương*** loại D với đồng bộ chung trên một dây. Như vậy, đầu vào xung đồng bộ được kích bằng xung chọn cổng ra có mức tích cực là logic 1 hoặc logic 0. Độ dốc của sườn xung phải đạt giá trị danh định (20ns).

Cả hai thanh chốt 74LS373 và 74LS374 đều có đầu ra ba trạng thái. Các bộ đệm ba trạng thái có đầu điều khiển ra với mức tích cực ở lôgíc 0. Nếu chúng được sử dụng như các cổng ra thì các bộ đệm được điều khiển bởi thiết bị ra.

5.3. VI MẠCH GHÉP NỐI CÓ LẬP TRÌNH 8255A

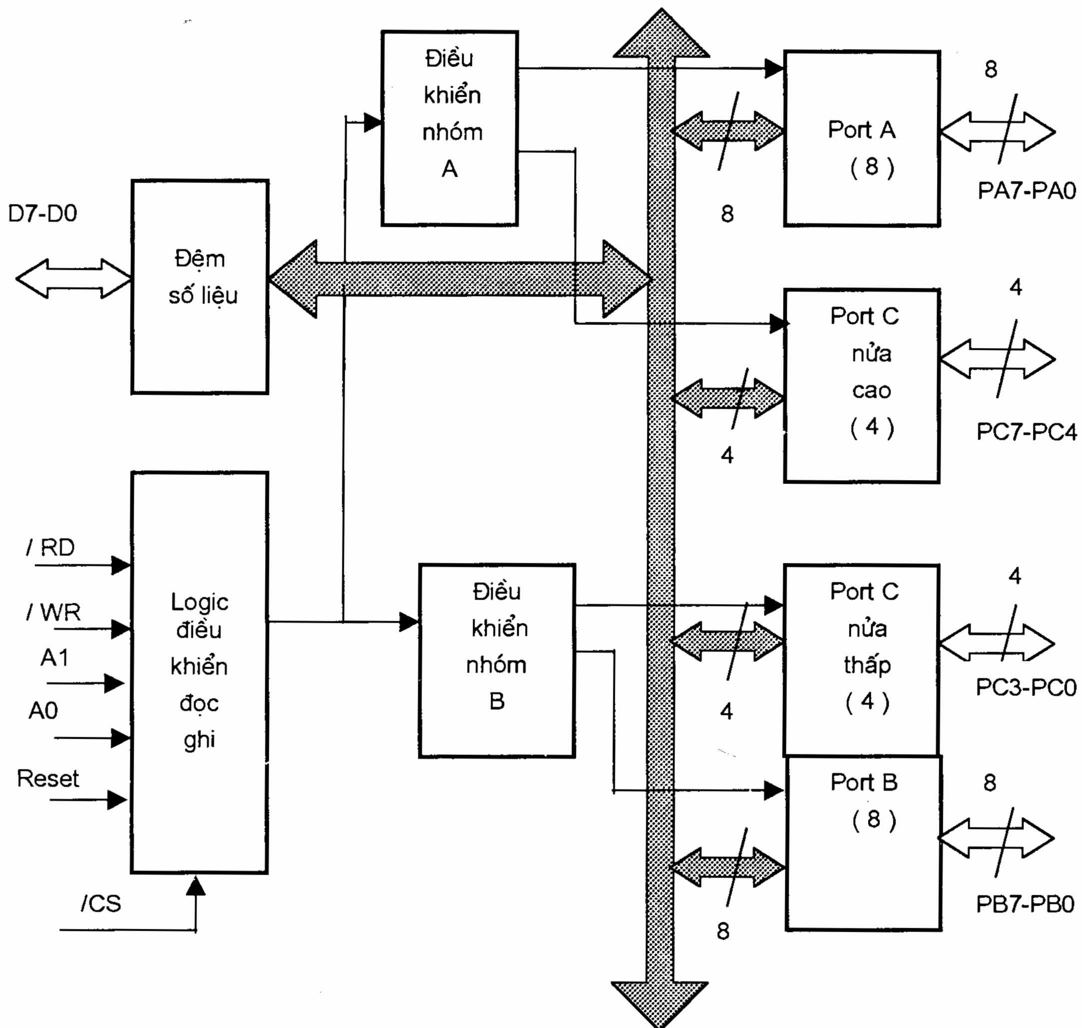
5.3.1. Cấu trúc của 8255A

Vi mạch 8255A là vi mạch cỡ lớn LSI, thường được gọi là mạch phối ghép vào/ra song song lập trình được (Programmable peripheral interface - PPI). Do khả năng mềm dẻo trong các ứng dụng thực tế, nó là mạch phối ghép được dùng rất phổ biến cho các hệ vi xử lý 8/16/32 biít. Vi mạch 40 chân này có các chân bố trí như ở trên hình 5.6, trong đó có chứa 24 bit vào/ra tạo thành ba cổng song song (portA, portB và portC). Một nửa của cổng portC (PC4 .... PC7) thuộc vào nhóm A, còn nửa kia thuộc nhóm B. Sơ đồ khối mô tả chức năng bên trong của 8255A được thể hiện trên hình 5.7.



Hình 5.6. **Tổ chức chân tín hiệu của 8255A**

Các chân tín hiệu của 8255A có chức năng rõ ràng. Chân Reset phải được nối với tín hiệu Reset chung của toàn hệ (khi reset thì các cổng được định nghĩa là cổng vào để không gây ra sự cố cho các mạch điều khiển).



***Hình 5.7.* Sơ đồ khối chức năng của 8255A**

Tín hiệu /CS được nối với mạch tạo xung chọn thiết bị để đặt mạch 8255A vào một địa chỉ cơ sở nào đó. Các tín hiệu địa chỉ A0, A1 sẽ chọn ra 4 thanh ghi bên trong 8255A: 1 thanh ghi để ghi từ điều khiển cho hoạt động của 8255A (viết tắt là CWR: control word register) và 3 thanh ghi khác ứng với các cổng (port) là PA, PB, PC để ghi/ đọc dữ liệu (bảng 5.1). Theo bảng này ta nhận thấy địa chỉ cho PA cũng chính là địa chỉ cơ sở của 8255A.

Tính linh hoạt của vi mạch này thể hiện ở khả năng lập trình. Qua một thanh ghi điều khiển, người sử dụng xác định Mode hoạt động và cổng nào cần được sử dụng như là lối vào hoặc lối ra. Các chân ra D0 - D7 tạo nên kênh dữ liệu hai hướng có độ rộng 8 bit. Tất cả các dữ liệu khi truy nhập ghi hoặc là đọc được dẫn qua kênh dữ liệu này.

Trạng thái lôgic ghi/ đọc được nhận biết qua các tín hiệu điều khiển /CS, /RD, /WR. Trao đổi thông tin với vi mạch 8255A chỉ có thể được tiến hành khi

/CS =0. Khi /RD =0, dữ liệu của cổng được chọn được đưa ra kênh dữ liệu và có thể được sử dụng bởi các vi mạch khác. Khi /WR =0, thì mọi việc xảy ra ngược lại. Các bit địa chỉ A0 và A1 cùng với các tín hiệu ghi và đọc báo hiệu cho biết sẽ truy nhập lên cổng nào.

Bảng 5.1

A1	A0	/CS	/RD	/WR	Lệnh (củaVXL)	Hướng chuyển số liệu (với VXL)
0	0	0	0	1	Đọc PortA	PortA → D0 ÷ D7
0	1	0	0	1	Đọc PortB	PortB → D0 ÷ D7
1	0	0	0	1	Đọc PortC	PortC → D0 ÷ D7
1	1	0	0	1		Không có giá trị
0	0	0	1	0	Ghi PortA	D0 ÷ D7 → PortA
0	1	0	1	0	Ghi PortB	D0 ÷ D7 → PortB
1	0	0	1	0	Ghi PortC	D0 ÷ D7 → PortC
1	1	0	1	0	Ghi thanh ghi điều khiển	D0 ÷ D7 → Thanh ghi điều khiển
x	x	1	x	x	Vi mạch ở trạng thái trở kháng cao	Không có trao đổi số liệu

Từ bảng 5.1 ta thấy rằng thanh ghi điều khiển có địa chỉ 03h (A1A0=11). Khi gán từ điều khiển vào thanh ghi này, có thể định nghĩa hướng cho cổng vào/ra cũng như Mode hoạt động của 8255A.

5.3.2. Các chế độ làm việc của 8255A

Có 2 loại từ điều khiển cho 8255A:

- Từ điều khiển định nghĩa cấu hình cho các cổng PA, PB, PC.
- Từ điều khiển lập/ xoá từng bit ở đầu ra của PC.

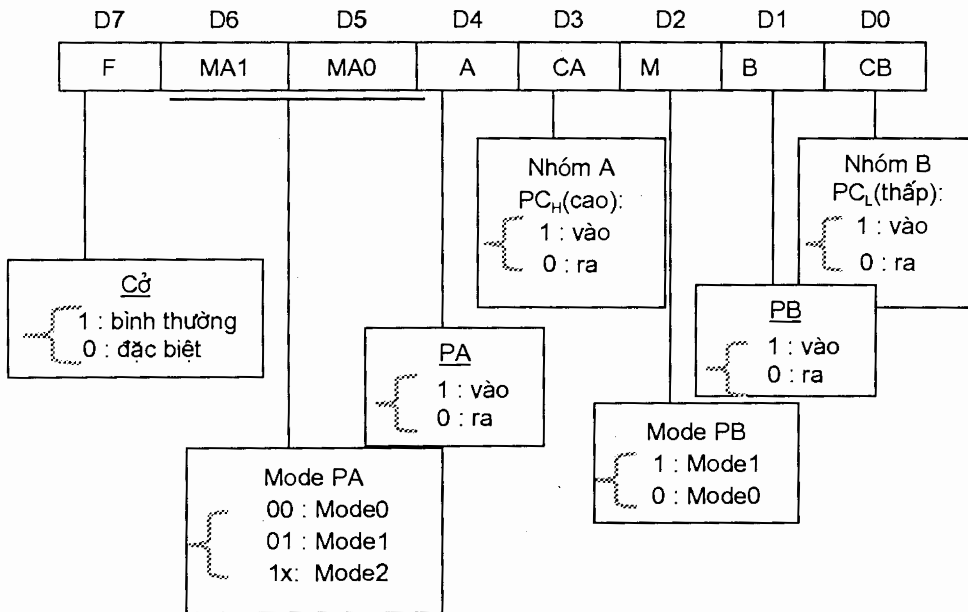
Tuỳ theo từ lệnh được ghi vào thanh ghi điều khiển khi khởi động của vi mạch mà ta có các PortA, B, C hoạt động ở :

Các chế độ 0, 1, 2 khác nhau.

Hướng trao đổi dữ liệu khác nhau, tức các PortA, B, C là cổng ra hay cổng vào.

Từ điều khiển định nghĩa cấu hình cho các cổng

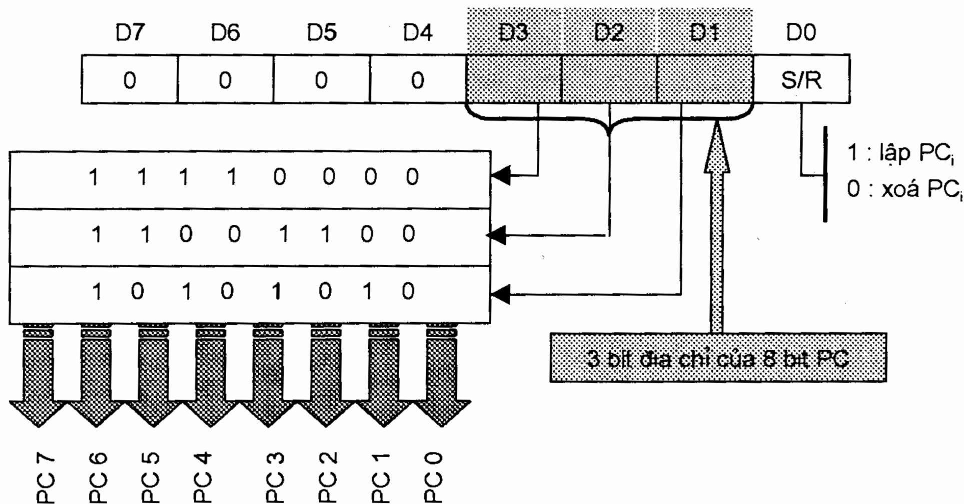
Dạng thức của từ điều khiển để định nghĩa cấu hình cho 8255A được thể hiện trên hình 5.8.



Hình 5.8. **Từ điều khiển cấu hình của 8255A**

Từ điều khiển lập xoá / bít ra PC_i (cờ F = 0)

Dạng thức của từ điều khiển để lập xoá PC_i được thể hiện trên hình 5.9.

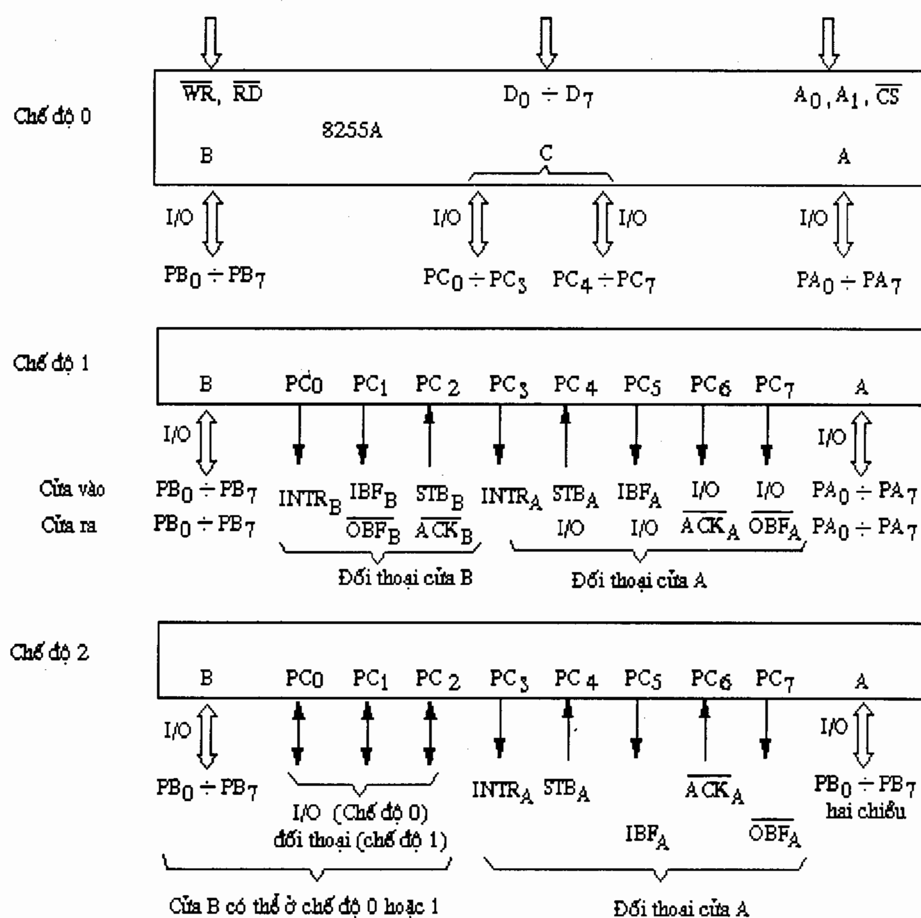


Hình 5.9. **Từ điều khiển để lập / xoá PC_i của 8255A**

Các chế độ (Mode) làm việc của mạch phối ghép 8255A có thể được định nghĩa bằng từ điều khiển CWR. Chip 8255A có 3 chế độ làm việc :

- Chế độ 0: Vào/ra cơ sở (còn gọi là vào ra đơn giản). Trong chế độ này mỗi cổng PA, PB, PC_H và PC_L đều có thể được định nghĩa là các cổng vào hoặc ra.
- Chế độ 1: Vào/ra có xung cho phép. Trong chế độ này mỗi cổng PA, PB có thể được định nghĩa thành cổng vào hoặc ra với các tín hiệu móc nối (handshaking) do các bit tương ứng của cổng PC trong cùng nhóm đảm nhận.
- Chế độ 2: Vào /ra 2 chiều. Trong chế độ này chỉ riêng cổng PA có thể được định nghĩa thành cổng vào/ra 2 chiều với các tín hiệu móc nối do các bit của cổng PC đảm nhiệm. Lúc này cổng PB có thể làm việc trong chế độ 0 hoặc 1.

Trong chế độ 0, người ta có thể dùng các bit của cửa C, tức các lối ra PC_i để lập (xác lập lên 1) và xoá (xoá về 0) để điều khiển hoặc đối thoại với thiết bị ngoại vi, chế độ này gọi là chế độ lập/xoá từng bit của cửa C, điều này có thể thấy rõ trên hình 5.9. Tóm tắt các chế độ làm việc của chip 8255A được thể hiện trên hình 5.10.



Hình 5.10. **Tóm tắt các chế độ làm việc của 8255A**

5.4. GHÉP NỐI 8255A VỚI HỆ VI XỬ LÝ

Trong chế độ 0, chip 8255 cho một khả năng xuất và nhập dữ liệu đơn giản qua 3 cổng A, B và C. Đối với chế độ này, từ điều khiển có thể tính theo cách sau:

	Port A	Port C Bit 4...7	Port B	Port C Bit 0 .. 3
Lối vào	16	8	2	1
Lối ra	0	0	0	0
Từ điều khiển:	128 + ☐	+ ☐	+ ☐	☐ +

Thí dụ 1: Kiểu 0, cổng A như là lối vào, cổng B như là lối ra, cổng C như là lối ra.

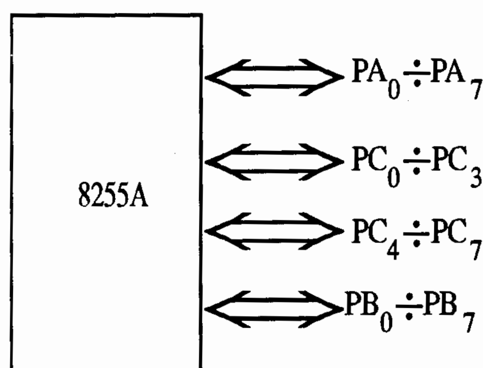
Từ điều khiển (Controlword) = 128 + 16 + 0 + 0 + 0 = 144.

Thí dụ 2: Kiểu 0, cổng A như là lối ra , cổng C (bit 4 ...7) như là lối vào, cổng B như là lối vào và cổng C (bit 0 ... 3) như là lối ra.

Từ điều khiển (Controlword) = 128 + 0 + 8 + 2 + 0 = 128.

Chip 8255A làm việc ở chế độ 0

Các cửa A, B và C được sử dụng độc lập với nhau. Ba tuyến PA, PB, PC đều được dùng để trao đổi số liệu một cách bình đẳng (hình 5.11).



***Hình 5.11.* Chip 8255A làm việc ở chế độ 0**

Chip 8255A làm việc ở chế độ 1

Là chế độ vào/ra có chốt (Strob), tức là có sự đối thoại giữa ngoại vi và hệ vi xử lý thông qua các bit của cổng PC. Có 2 nhóm:

Nhóm A: Gồm PortA dùng để trao đổi số liệu và nửa cao PortC ($PC_4 \div PC_7$) để đối thoại giữa vi xử lý và ngoại vi.

+ Nhóm B: Gồm PortB để trao đổi số liệu và nửa thấp PortC ($PC_0 \div PC_3$) để đối thoại giữa vi xử lý và ngoại vi.

Hướng và chế độ 1 của PortA, PortB do từ điều kiển quyết định, các tín hiệu đối thoại PC_i phụ thuộc hướng cổng vào hay ra.

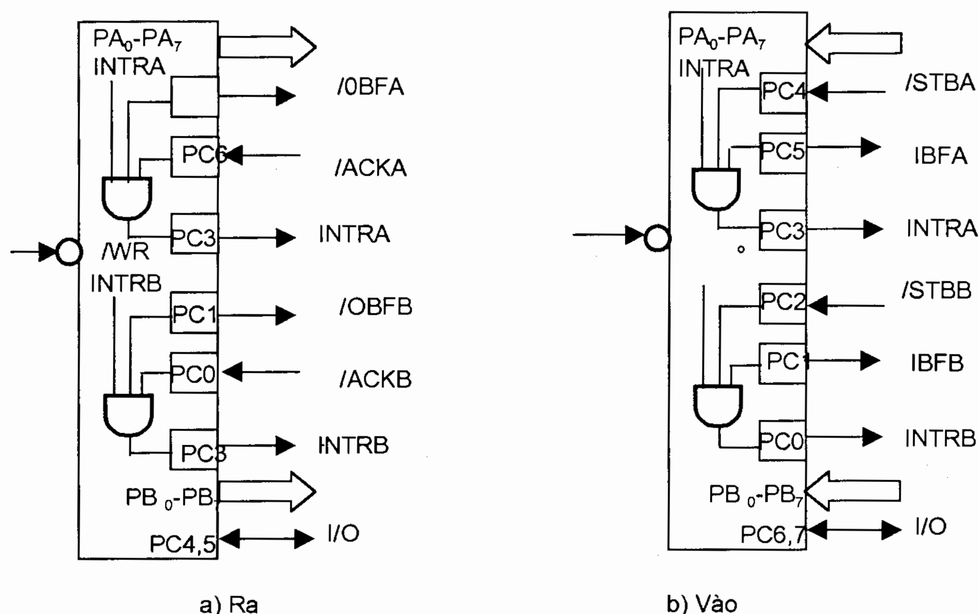
Sơ đồ ghép nối của 8255A ở Mode 1 và tín hiệu đối thoại được thể hiện trên hình 5.12.

♦ Xuất dữ liệu ra trong Mode 1

Các cổng PA, PB có tín hiệu đối thoại tương tự nhau. Tín hiệu /OBFA, /OBFB báo rằng bộ đệm ra đã đầy cho ngoại vi biết là CPU đã ghi dữ liệu vào cổng để chuẩn bị đưa ra. Tín hiệu này thường nối với tín hiệu /STR của thiết bị nhận.

Tín hiệu /ACKA và /ACKB là tín hiệu của ngoại vi cho biết nó đã nhận được dữ liệu từ các cổng PA, PB.

Tín hiệu INTRA (INTRB) là tín hiệu yêu cầu ngắt từ PA (PB). Đây là kết quả thu được từ quan hệ giữa các tín hiệu khác của 8255A trong quá trình đối thoại với ngoại vi. Nó được dùng để phản ảnh yêu cầu ngắt của PA(PB) tới CPU.



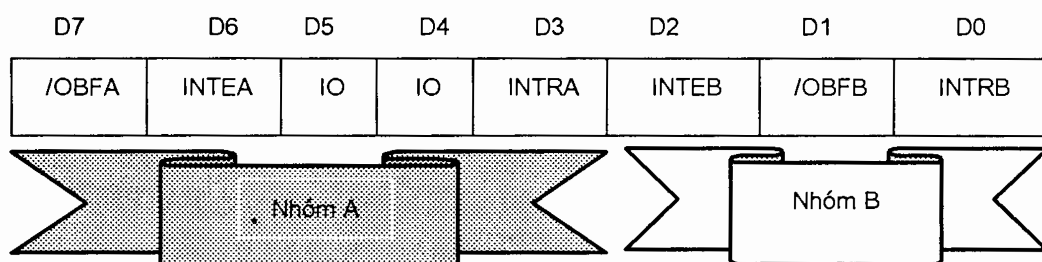
Hình 5.12. **Chip 8255A làm việc ở chế độ 1**

INTEA (INTEB) là tín hiệu của một mạch lật bên trong 8255A để cho phép hoặc cấm yêu cầu ngắt INTRA của PA (hay INTRB của PB).

INTEA được lập/xoá thông qua bit PC_6 của PC_i .

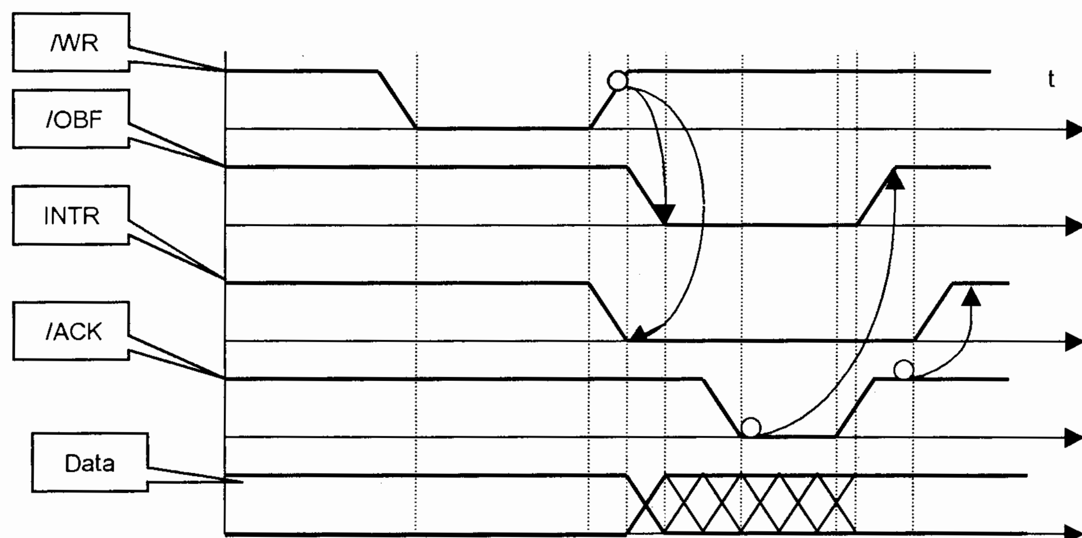
INTEB được lập xoá thông qua bit PC_2 của PortC.

Khi làm việc ở chế độ xuất thông tin Mode 1, thanh ghi trạng thái của 8255 cung cấp các thông tin phản ánh trạng thái hiện hành của mình. Nếu hệ vi xử lý cần những thông tin này thì chỉ cần đọc vào để thẩm định. Nội dung thanh ghi trạng thái được thể hiện trên hình 5.13.



Hình 5.13. **Nội dung thanh ghi trạng thái của 8255 ở Mode 1 cho hướng ra.**

Hoạt động của 8255 khi xuất thông tin ra cho ngoại vi ở Mode 1 được mô tả bằng đồ thị thời gian hình 5.14.



Hình 5.14. **Đồ thị thời gian của 8255 khi xuất thông tin ra cho ngoại vi ở Mode 1**

Giải thích tên và chức năng tín hiệu trong thanh ghi trạng thái của 8255.

/OBFA (Output Buffer A full) – cổng A có dữ liệu rồi.

/OBFB (Output Buffer B full) – cổng B có dữ liệu rồi.

INTEA (Interrupt Enable for portA) cho phép cổng A chạy ở chế độ ngắt.

INTEB (Interrupt Enable for portB) cho phép cổng B chạy ở chế độ ngắt.

INTRA (Interrupt portA) - cổng A ngắt.

INTRB (Interrupt portB) - cổng B ngắt.

IO Hướng truyền của PC4 và PC5.

♦ Nhận dữ liệu vào trong Mode 1

Khi nhận dữ liệu vào trong Mode 1 các cổng PA, PB có tín hiệu đối thoại tương tự nhau. Các tín hiệu đó là:

/STB (cho phép chốt dữ liệu). Khi dữ liệu đã sẵn sàng trên kênh cho PA (PB), ngoại vi phải dùng /STB để báo cho 8255A biết để chốt dữ liệu vào cổng PA (hay PB).

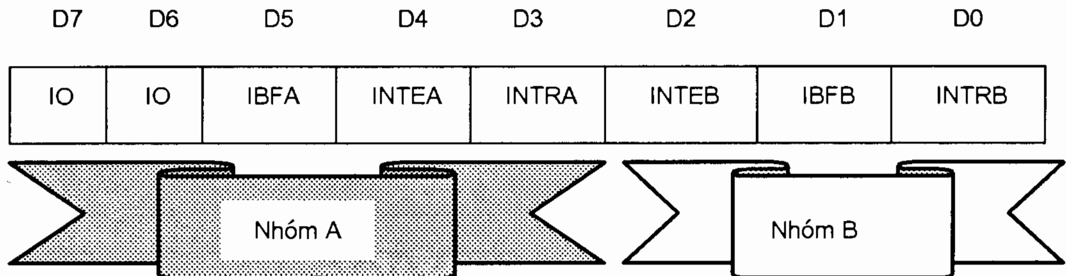
IBF (In Buffer full - đệm vào đầy). Sau khi 8255A chốt được dữ liệu do thiết bị ngoại vi đưa đến, nó đưa ra tín hiệu IBF để báo cho ngoại vi biết đã chốt xong. Tín hiệu INTRA (INTRB) là tín hiệu yêu cầu ngắt từ cổng PA (PB) - báo cho CPU biết đã có dữ liệu trong PA (PB) để CPU đọc vào hệ vi xử lý.

INTEA (INTEB) là tín hiệu của một mạch lật bên trong 8255A để cho phép hoặc cấm yêu cầu ngắt INTRA của PA (INTRB của PB).

INTEA được lập/xoá thông qua bit PC_4 của PortC.

INTEB được lập/xoá thông qua bit PC_2 của PortC.

Khi làm việc ở chế độ nhận thông tin của Mode 1, thanh ghi trạng thái của 8255 cung cấp các thông tin phản ánh trạng thái hiện hành của mình. Nếu hệ vi xử lý cần những thông tin này thì chỉ cần đọc vào để thẩm định. Nội dung thanh ghi trạng thái được thể hiện trên hình 5.15.



***Hình 5.15.* Nội dung thanh ghi trạng thái của 8255 ở Mode 1 cho hướng vào.**

Giải thích tên và chức năng tín hiệu trong thanh ghi trạng thái của 8255.

IBFA (Input Buffer A full) – cổng A có dữ liệu rồi.

IBFB (Input Buffer B full) – cổng B có dữ liệu rồi.

INTEA (Interrupt Enable for portA) cho phép cổng A chạy ở chế độ ngắt.

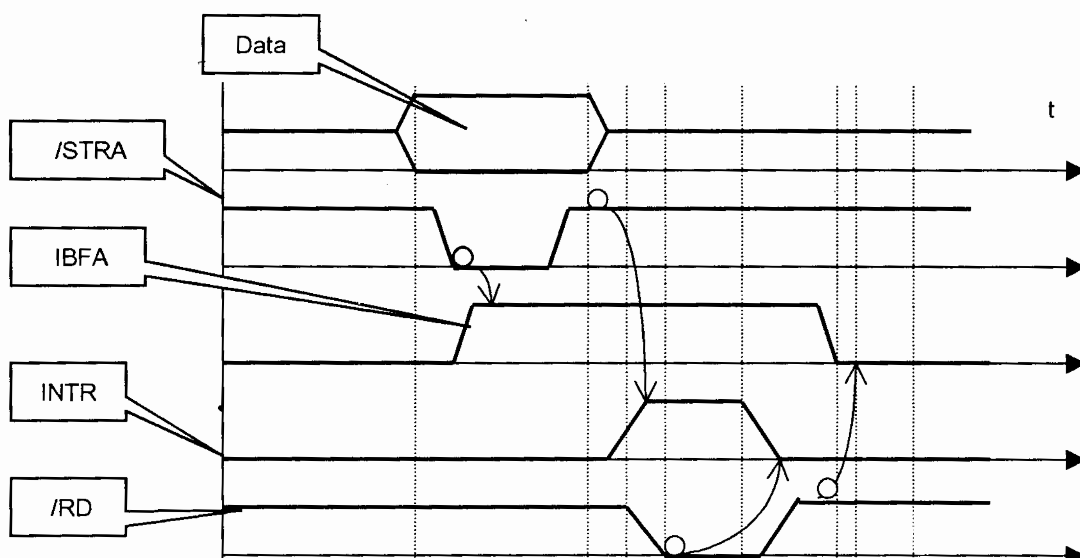
INTEB (Interrupt Enable for portB) cho phép cổng B chạy ở chế độ ngắt.

INTRA (Interrupt portA) - cổng A ngắt.

INTRB (Interrupt portB) - cổng B ngắt.

IO Hướng truyền của PC6 và PC7.

Hoạt động của 8255 khi nhận thông tin vào từ ngoại vi ở Mode 1 được mô tả bằng đồ thị thời gian hình 5.16.



Hình 5.16. **Đồ thị thời gian của 8255 khi nhận thông tin vào từ ngoại vi ở Mode 1.**

Chip 8255A làm việc ở chế độ 2

Chế độ này chỉ dùng cho cổng PA với vào/ra hai chiều, các bit PC_3, PC_4 đến PC_7 dùng làm tín hiệu đối thoại.

Mạch logic của 8255A ở Mode 2 và các tín hiệu đối thoại được thể hiện trên hình 5.18.

Cổng PB có thể làm việc ở Mode 1 hoặc Mode 0 tuỳ theo các bit điều khiển trong thanh ghi CWR.

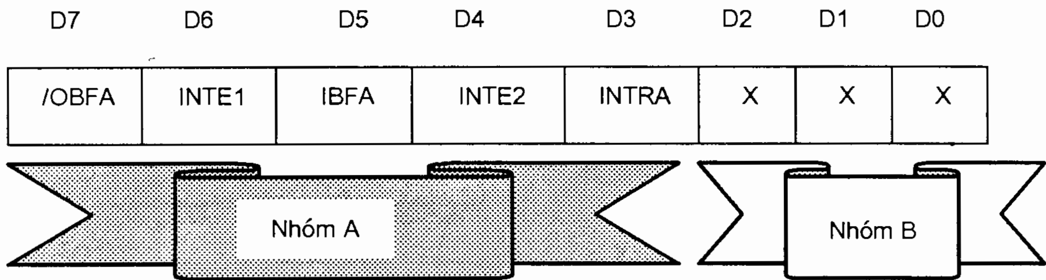
+ INTRA: Yêu cầu ngắt cho dữ liệu hai chiều vào/ra.
+ INTE1, INTE2: Là hai tín hiệu của hai mạch lật bên trong 8255A để cho phép hoặc cấm yêu cầu ngắt của PA, các bit này được lập xoá bởi PC_6 và PC_4 của PC.

Khi dùng 8255A trong chế độ kênh hai chiều để trao đổi dữ liệu theo cách thăm dò, phải kiểm tra xem bit /OBFA có bằng 1 (đệm ra rỗng) hay không trước khi dùng lệnh Out để đưa dữ liệu ra cổng PA, hoặc kiểm tra xem bit IBFA có bằng 0 (đệm vào rỗng) hay không trước khi dùng lệnh IN để nhận dữ liệu vào từ cổng PA.

PA: Mode 2	PA: Mode 2
PB: Mode 0 (vào)	PB: Mode 1 ra.

$PC_0 \div PC_2$: I/O tuỳ từ điều khiển.

Khi làm việc ở chế độ truyền thông tin hai chiều của Mode 2, thanh ghi trạng thái của 8255 cung cấp các thông tin phản ánh trạng thái hiện hành của mình. Nếu hệ vi xử lý cần những thông tin này thì chỉ cần đọc vào để thẩm định. Nội dung thanh ghi trạng thái được thể hiện trên hình 5.17.



Hình 5.17. **Nội dung thanh ghi trạng thái của 8255 ở Mode 2 cho hướng 2 chiều.**

Giải thích tên và chức năng tín hiệu trong thanh ghi trạng thái của 8255.

IBFA (Input Buffer A full) – cổng A có dữ liệu vào rồi.

/OBFB (Output Buffer A full) – cổng A có dữ liệu ra rồi.

INTE1 (Interrupt Enable1 for portA) cho phép cổng A chạy ở chế độ ngắt.

INTE2 (Interrupt Enable2 for portA) cho phép cổng A chạy ở chế độ ngắt.

INTRA (Interrupt portA) – cổng A ngắt.

XXX Định nghĩa Mode0 hay Mode1 cho cổng PortB.

Hình 5.18. **Chip 8255A làm việc ở chế độ 2: PortA 2 chiều còn**
a) PortB vào; b) PortB ra.

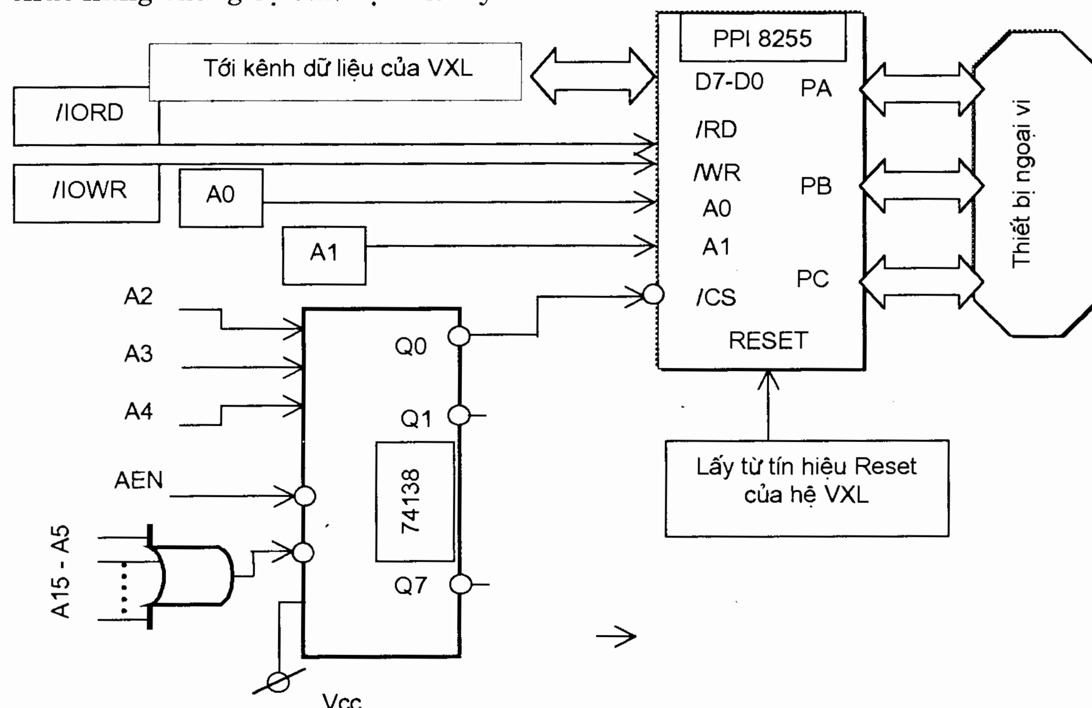
Một phương án ghép nối chip 8255 với hệ vi xử lý và thiết bị ngoại vi ở Mode 0 được trình bầy trên hình 5.19. Phần giải mã chọn chip được thực hiện trên IC 74138. Kênh địa chỉ của 74138 có 3 bit ABC được nối tương ứng với các bit A2A3A4 của kênh địa chỉ của hệ vi xử lý. Chân E1 của 74138 được nối +Vcc. Chân /E3 của 74138 được nối tín hiệu AEN còn chân /E2 là đầu ra của tổ hợp A15-A5 theo mạch OR. Với tổ chức của 74138 như vậy, tín hiệu chip select /CS cho 8255 là 0000h, 0001h, 0002h, 0003h. Những địa chỉ này chính là các thanh ghi bên trong của 8255 nếu ta sử dụng 2 bit A0 và A1 của hệ vi xử lý ghép với A0 và A1 của chip 8255. Như vậy: PortA của 8255 có địa chỉ là 0000h.

PortB của 8255 có địa chỉ là 0001h.

PortC của 8255 có địa chỉ là 0002h.

Thanh ghi lệnh/trạng thái của 8255 có địa chỉ là 0003h.

Kênh dữ liệu của chip 8255 D7-D0 được nối với byte thấp của kênh dữ liệu hệ vi xử lý. Các tín hiệu điều khiển /RD, /WR, Reset được nối với các tín hiệu có chức năng tương tự của hệ vi xử lý.



Hình 5.19. **Ghép nối chip vào/ra 8255 với hệ VXL**

Ba cổng PortA, PortB, PortC được nối ghép với thiết bị ngoại vi. Tuỳ theo chức năng hoạt động mà các cổng này có thể lập trình điều khiển hướng truyền tin là ra hay vào.

Ví dụ 1: Cần nhận thông tin từ ngoại vi 1 nối với cổng PA của 8255 từng byte rồi xuất ngay ra cho ngoại vi 2 nối với cổng PB. Biết rằng số byte cần chuyển là 100.

Như vậy hệ vi xử lý đóng vai trung chuyển dữ liệu từ ngoại vi 1 sang cho ngoại vi 2. Sử dụng sơ đồ ghép nối trên hình 5.19 dễ dàng lập được byte lệnh điều khiển cho hoạt động của 8255. Cụ thể ở phần khai báo ta đặt tên gợi nhớ và gán giá trị đúng cho chúng:

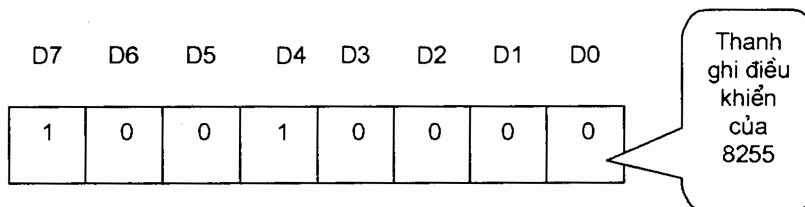
PortA EQU 00h

PortB EQU 01h

PortC EQU 02h

CWR EQU 03h

Lập byte điều khiển cho thanh ghi lệnh CWR:



Vậy là byte điều khiển có giá tri 90h. Ta dùng lệnh gán

CW EQU 90h.

Đoạn chương trình sau thực hiện chức năng yêu cầu:

MOV AL, CW; AL nhận byte điều khiển

OUT CWR,CW; nạp byte này vào thanh ghi lệnh của 8255

MOV CX,100; sử dụng CX làm bộ đếm 100 (sẽ dùng lệnh loop)

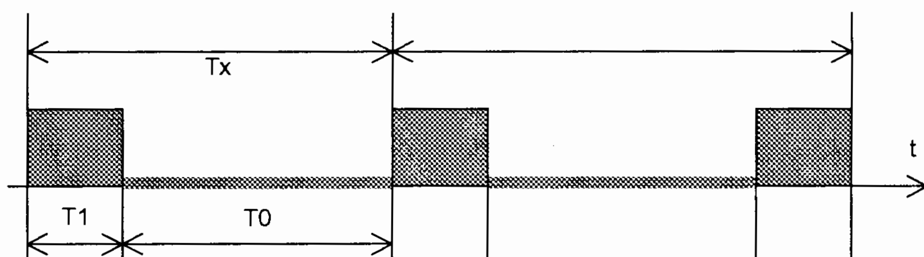
LOOP100:

IN AL,PORTA ;nhận byte dữ liệu từ portA

OUT PORTB,AL;chuyển ngay ra portB

LOOP LOOP100; lặp lại 100 lần.

Ví dụ 2: Cần tạo tín hiệu xung trên bit PC0 (portC của 8255) có chu kỳ lặp lại theo dạng sau: Độ rộng theo thời gian : T1+T0=Tx và T0=2*T1.



Theo yêu cầu của nhiệm vụ thì cổng C phải là cổng ra cho nên nếu sử dụng sơ đồ 5.19 để thực hiện chức năng tạo xung thì khai báo byte lệnh có thay đổi có thay đổi. Byte lệnh có giá trị 00h cho giá trị 0 ở lối ra PC0 và 01h cho giá trị 1 ở lối ra PC0.

```
CWR  EQU 03h
CW1  EQU 01h
CW0  EQU 00h
```

Đoạn chương trình sau sẽ thực hiện chức năng yêu cầu:

```
LOOP_PULSE:
        MOV AL,CW1; AL nhận byte điều khiển (PC 0=1)
        OUT CWR,AL; nạp byte này vào thanh ghi lệnh của 8255
        CALL Delay1;
        MOV AL,CW0; AL nhận byte điều khiển (PC 0=0)
        OUT CWR,AL; nạp byte này vào thanh ghi lệnh của 8255
        CALL Delay0;
        JMP  LOOP_PULSE; lặp lại
```

Chu kỳ xung nhịp gồm xung logic 1 có độ rộng bằng đúng thời gian thực hiện của chương trình con delay1 và xung logic 0 có độ rộng bằng đúng thời gian thực hiện của chương trình con delay0. Quá trình này là quá trình không có điểm dừng. Mối quan hệ thời gian giữa delay1 và delay0 dễ dàng thực hiện bằng phần mềm. Đoạn chương trình sau cho phép tạo độ rộng xung tuỳ ý cho xung logic 1:

```
        MOV CX,n ; số đếm đặt trong CX
  LOOP1:  ;vòng lặp
        MOV SI, m ; số đếm đặt trong SI
        LOOP_SI: ; vòng lặp lồng trong vòng lặp trên
              DEC SI
              JNZ LOOP_SI
        LOOP LOOP1
```

Với 2 vòng lặp và hai tham số điều chỉnh như vậy, rất dễ trong điều chỉnh thời gian cho xung logic1. Để tạo delay0 chỉ cần gọi 2 lần chương trình con delay1:

```
        CALL delay1
        CALL delay1
```

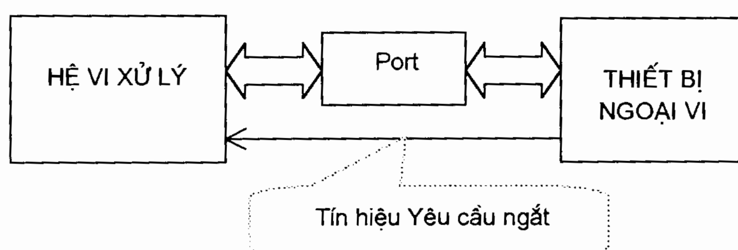
là đã có độ rộng của xung logic 0 bằng 2 lần độ rộng xung logic 1.

Chương 6
CHẾ ĐỘ NGẮT CỦA BỘ VI XỬ LÝ

6.1. CHẾ ĐỘ NGẮT CỦA BỘ VI XỬ LÝ

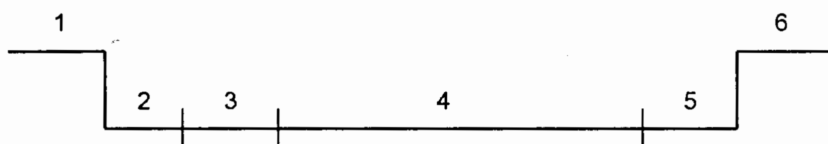
Chế độ ngắt là chế độ đặc biệt mà bất kỳ bộ vi xử lý nào cũng được trang bị để tạo cơ chế điều khiển mềm dẻo và linh hoạt khi hệ có nhiều thiết bị ngoại vi được ghép nối và hoạt động. Bản thân chế độ ngắt là chế độ dị bộ, nghĩa là các yêu cầu được phục vụ từ phía ngoại vi tới hệ vi xử lý là không xác định về mặt thời điểm xuất hiện.

Như vậy bộ vi xử lý có thể tiến hành công việc chính của mình, chỉ khi nào xuất hiện yêu cầu trao đổi dữ liệu từ phía ngoại vi thì bộ vi xử lý mới tạm dừng công việc hiện tại để phục vụ việc trao đổi dữ liệu với ngoại vi đó. Sau khi hoàn thành việc trao đổi dữ liệu thì bộ vi xử lý lại phải quay về được để làm tiếp công việc đã bị gián đoạn. Cách làm việc theo kiểu này gọi là ngắt CPU để trao đổi dữ liệu. Cơ chế này được minh hoạ trên hình 6.1.



***Hình 6.1.* Hoạt động của cơ chế ngắt trong hệ vi xử lý**

Khi hoạt động ở cơ chế ngắt thiết bị ngoại vi chủ động phát tín hiệu ngắt cho hệ vi xử lý qua chân INT của CPU. CPU có phương thức kiểm tra sự xuất hiện của tín hiệu yêu cầu ngắt này và sẽ đáp ứng khi có thể. Khi một ngắt nào đó có hiệu lực (tức là ngắt được CPU chấp nhận) thì một trình tự các thao tác như sau được thực hiện một cách tự động (hình 6.2).



Hình 6.2. **Thao tác 6 bước thực hiện ngắt của CPU**

Giải thích thao tác:

a) Trường hợp CPU là bộ vi xử lý 16/32 bit

Nếu ngắt có số hiệu N kích hoạt và có hiệu lực.

Bước 1: CPU kết thúc lệnh đang thực hiện.

Bước 2: CPU thực hiện các thao tác cất giữ thông tin trạng thái mà thông tin quan trọng nhất là giá trị con trỏ chương trình CS: IP rồi mới chuyển điều khiển tới chương trình con phục vụ ngắt. Cụ thể

(SP) ← (SP)-2 chuẩn bị chứa thanh ghi cờ 16 bit

((SP)) ←(F) trong đó ((SP)) là ô nhớ do (SP) chỉ ra.

(IF)← 0 . (TF) ← 0

(cấm các ngắt khác tác động vào CPU, cho CPU chạy ở chế độ bình thường)

(SP)← (SP)-2

((SP))← (CS)

(Chỉ ra đỉnh mới của ngăn xếp, cất địa chỉ Seg của địa chỉ trở về vào đỉnh ngăn xếp).

(SP)← (SP)-2

((SP))← (IP)

(Chỉ ra đỉnh mới của ngăn xếp, cất địa chỉ offset của địa chỉ trở về vào đỉnh ngăn xếp.

Chuyển điều khiển tới điểm vào của chương trình con phục vụ ngắt bằng cách gán địa chỉ của chương trình con phục vụ ngắt số hiệu N tương ứng trong bảng vectơ ngắt vào cặp thanh ghi CS:IP (cho trường hợp cấp phát 4byte/1vector ngắt), tức là:

((N*4)) → (IP) và ((N* 4+2)) → (CS)

Bước 3: Cất giữ nội dung các thanh ghi đa năng khác (nếu cần) vào ngăn xếp.

Bước 4: Thực hiện chương trình con.

Bước 5: Khôi phục trạng thái cũ của các thanh ghi đa năng.

Bước 6: Kết thúc chương trình con, trao quyền điều khiển cho chương trình chính khi gặp lệnh IRET. Cụ thể:

(IP)←((SP)) ; chuyển 2 ngăn nhớ liên tục của ngăn xếp vào IP

(SP)← (SP) +2; con trỏ ngăn xếp tăng 2 đơn vị.

(CS)←((SP)) ; chuyển 2 ngăn nhớ liên tục của ngăn xếp vào CS

(SP)← (SP) +2; con trỏ ngăn xếp tăng 2 đơn vị.

(F) ←((SP)) ; chuyển 2 ngăn nhớ liên tục của ngăn xếp vào thanh ghi cờ.

(SP) ← (SP) +2; con trỏ ngăn xếp tăng 2 đơn vị.

Bộ vi xử lý quay lại chương trình chính tại địa chỉ trở về và với giá trị cũ của thanh ghi cờ được lấy ra từ ngăn xếp.

b) Trường hợp CPU là bộ vi xử lý 8 bit (như 8085Intel chẳng hạn).

Nếu ngắt có số hiệu N kích hoạt.

Bước 1: CPU kết thúc lệnh đang thực hiện.

Bước 2: CPU thực hiện các thao tác cất giữ thông tin trạng thái mà thông tin quan trọng nhất là giá trị con trỏ chương trình PC rồi mới chuyển điều khiển tới chương trình con phục vụ ngắt. Cụ thể

(SP)← (SP)-1

((SP))← (PCh) ;byte cao của con trỏ chương trình PC cất vào ngăn xếp.

(SP)← (SP)-1

((SP))← (SPl) ;byte thấp của con trỏ chương trình PC cất vào ngăn xếp.

Chuyển điều khiển tới điểm vào của chương trình con phục vụ ngắt bằng cách gán địa chỉ của chương trình con phục vụ ngắt số hiệu N tương ứng trong bảng vectơ ngắt vào con trỏ chương trình PC (8 byte/1vector ngắt), tức là:

((N*8)) → (PCl) và ((N* 8+1)) → (PCh)

Bước 3: Cất giữ nội dung các thanh ghi đa năng khác (nếu cần) vào ngăn xếp.

Bước 4: Thực hiện chương trình con.

Bước 5: Khôi phục trạng thái cũ của các thanh ghi đa năng.

Bước 6: Kết thúc chương trình con, trao quyền điều khiển cho chương trình chính khi gặp lệnh RET. Cụ thể:

(PCl)←((SP)) ; chuyển byte thấp ngăn xếp do SP trỏ tới vào PCl

(SP)← (SP) +1; con trỏ ngăn xếp tăng 1 đơn vị.

(PCh)←((SP)) ; chuyển byte cao ngăn xếp do SP trỏ tới vào PCh

(SP) ← (SP) +1; con trỏ ngăn xếp tăng 1 đơn vị.

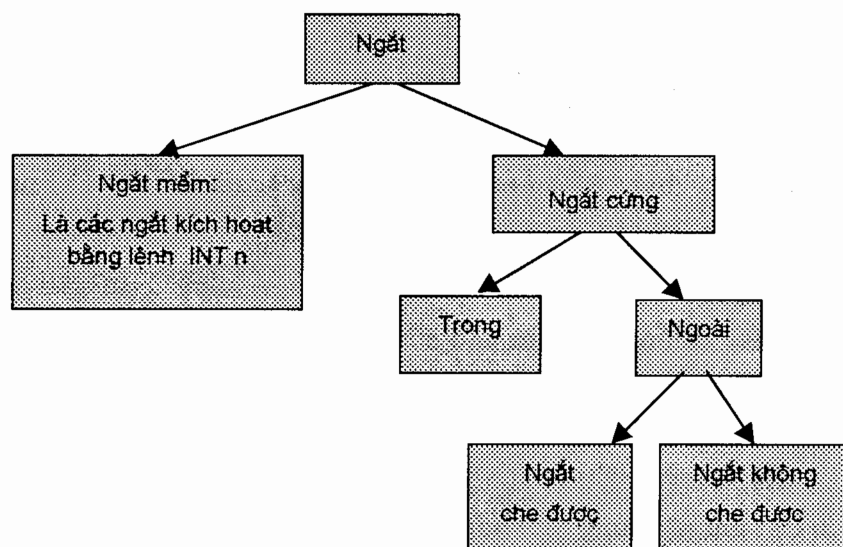
Một hệ vi xử lý hoạt động theo cơ chế ngắt có thể đáp ứng nhanh các yêu cầu trao đổi dữ liệu với ngoại vi trong khi vẫn có thể làm được các công việc khác. Muốn đạt được hiệu quả này hệ phải có cách tổ chức các chương trình phục vụ ngắt thành thư viện và kích hoạt chúng thông qua số hiệu ngắt chứ

không qua địa chỉ trực tiếp. Các chân tín hiệu cho các yêu cầu ngắt che được INTR và không che được NMI sẽ được sử dụng vào việc đưa các yêu cầu ngắt theo số hiệu từ bên ngoài đến CPU.

6.2. TỔ CHỨC NGẮT TRONG HỆ VI XỬ LÝ 80X86

6.2.1. Phân loại ngắt

Trong phần trước, chúng ta mới chỉ quan tâm đến các ngắt về mặt nguyên tắc, mà không phân biệt các loại ngắt. Các chương trình con phục vụ ngắt trong thư viện phần mềm của hệ vi xử lý có thể kích hoạt bằng hai phương thức cơ bản là dùng lệnh ngắt (ngắt mềm) và dùng phần cứng (ngắt cứng). Hình 6.3 giới thiệu các loại ngắt này:



Hình 6.3. **Các kiểu ngắt của hệ vi xử lý**

Ngắt mềm

Ngắt mềm là ngắt được gọi bằng một lệnh ở trong chương trình ngôn ngữ máy. Lệnh đó là INT, nó được sử dụng cùng với số hiệu ngắt. Thí dụ, lệnh gọi ngắt số 5 được viết là INT 5. Các ngắt mềm cho phép gọi trực tiếp các chương trình con phục vụ ngắt chứa trong thư viện chương trình. Điều này không những thực hiện được ở mức ngôn ngữ Assembly, mà còn bằng cả ngôn ngữ bậc cao. Tuy nhiên phải hiểu rằng lệnh gọi ngắt từ ngôn ngữ bậc cao thì trình biên dịch sau đó sẽ phải dịch ra thành lệnh Assembly INT, vì chỉ có lệnh này mới cho phép khởi động ngắt mềm. Do vậy, lệnh ngắt mềm bản chất là lệnh gọi CALL đặc biệt. Nó được gọi một cách chủ động bằng chương trình của người lập trình.

Ngắt cứng

Là các yêu cầu ngắt CPU do các tín hiệu đến từ các chân INTR và NMI. Khác với ngắt mềm, ngắt cứng không được khởi động bởi chương trình mà bởi các thành phần có trong phần cứng của hệ vi xử lý (thí dụ như các thiết bị ngoại vi trong và ngoài). Loại ngắt này là một cơ cấu đơn giản và hiệu quả để bộ xử lý phản ứng kịp thời với các sự kiện không đồng bộ xẩy ra trong hệ vi xử lý.

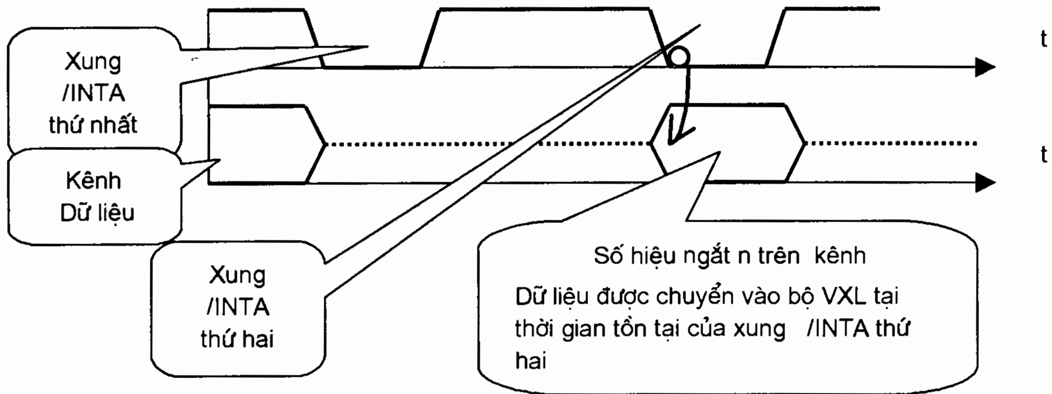
Điều này được minh hoạ thông qua thí dụ về cơ chế hoạt động của ngắt bàn phím nếu ngoại vi này được tổ chức giao tiếp với hệ vi xử lý theo cơ chế ngắt. Mỗi khi ấn hay nhả một phím thì ngắt bàn phím sẽ được kích hoạt. Chương trình xử lý ngắt sẽ được khởi động bằng cách chuyển kí tự được ấn vào vùng bộ đệm của bàn phím, xếp ngay sau kí tự được ấn lần trước. Cũng như mọi ngắt khác, việc thực hiện chương trình chính sẽ được khôi phục ngay sau khi chương trình con xử lý ngắt bàn phím kết thúc.

Với ví dụ trên thì ngắt bàn phím là ngắt cứng ngoài, vì nó được kích hoạt bởi thiết bị ngoại vi ở bên ngoài. Đối với loại ngắt này, chúng ta còn phân biệt ngắt che (cấm) được và không che được. Các ngắt này bị cấm bằng lệnh Assembly CLI (CLear Interrupt Flag). Khi một ngắt bị cấm, thì mặc dù được gọi, chương trình con tương ứng cũng không được thực hiện. Trong trường hợp bàn phím, điều đó có nghĩa là không ký tự nào được vào từ bàn phím nữa. Để huỷ bỏ chế độ cấm ngắt, người ta dùng lệnh STI (Set Interrupt Flag), nó cho phép các ngắt bị cấm trở lại hoạt động bình thường.

Các lệnh CLI và STI có ảnh hưởng trực tiếp tới trạng thái cờ IF trong bộ vi xử lý, tức là ảnh hưởng tới việc CPU có nhận biết yêu cầu ngắt tại chân INTR hay không. Yêu cầu ngắt tại chân INTR có thể có số hiệu *n* nằm trong khoảng 0-FFH. Kiểu ngắt này phải được đưa vào kênh dữ liệu để CPU có thể đọc được khi có xung INTA trong chu kỳ trả lời chấp nhận ngắt.

Các ngắt không thể bị cấm sẽ luôn được thực hiện, kể cả khi ngắt này được gọi ngay sau lệnh CLI. Thí dụ, ngắt qua chân NMI là ngắt loại này. Chương trình con phục vụ ngắt này thường là chương trình có chức năng thông báo các sự kiện quan trọng nhất đối với hệ như sự cố nguồn nuôi, sự cố thời gian thực...

Loại ngắt cuối cùng mà chúng ta cần có khái niệm là ngắt cứng nội bộ (trong). Những ngắt này không bị kích hoạt bởi thiết bị nằm bên ngoài hệ, mà bởi các chip IC bổ trợ nằm ngay trong hệ như ngắt thời gian khi nó được gọi bởi bộ tạo thời gian thực TIMER LSI 8253.

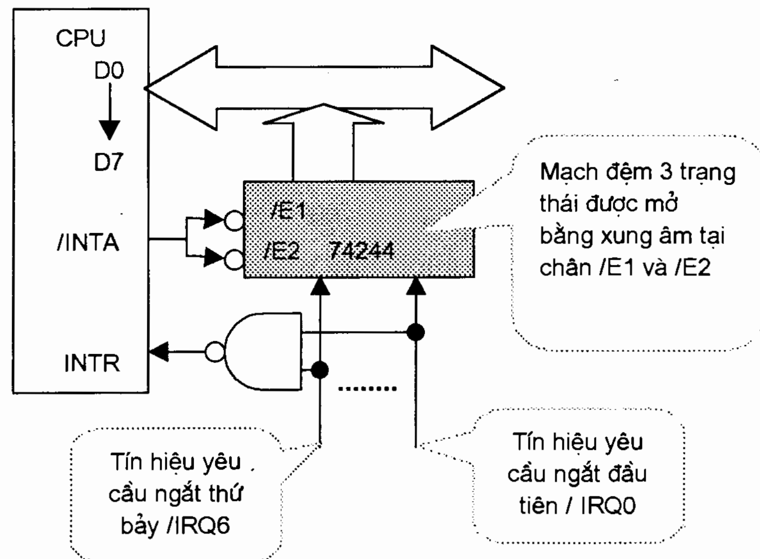


Hình 6.4. **Chu kỳ trả lời ngắt INTA của bộ vi xử lý**

6.2.2. Hoạt động của ngắt

Hoạt động của ngắt, đặc biệt là hoạt động của ngắt cứng có mối quan hệ chặt chẽ với các tín hiệu điều khiển của bộ vi xử lý như tín hiệu INR, /INTA, NMI. Ngoại vi có yêu cầu ngắt phải phối hợp các tín hiệu của mình với các tín hiệu của bộ vi xử lý để thực hiện cơ chế này theo đúng trình tự yêu cầu. Đồ thị thời gian của quá trình ngắt được mô tả trên hình 6.4 là phương thức tối thiểu phải được thực hiện.

Trên hình 6.5 mô tả sơ đồ đơn giản để đưa được số hiệu ngắt n vào kênh dữ liệu trong khi cùng tạo ra yêu cầu ngắt đưa vào chân INTR của bộ vi xử lý.



Hình 6.5. **Đưa số hiệu ngắt vào kênh dữ liệu**

Bảng 7.1

AD7	IR6	IR5	IR4	IR3	IR2	IR1	IR0	n
1	1	1	1	1	1	1	0	FEh(254)
1	1	1	1	1	1	0	1	FDh(253)
1	1	1	1	1	0	1	1	FBh(251)
1	1	1	1	0	1	1	1	F7h(247)
1	1	1	0	1	1	1	1	EFh(239)
1	1	0	1	1	1	1	1	DFh(223)
1	0	1	1	1	1	1	1	BFh(191)

Giả sử trong một thời điểm nhất định chỉ có một yêu cầu ngắt IRi được tác động và khi đó ở đầu ra của mạch NAND sẽ có xung yêu cầu ngắt đến CPU. Đồng thời tín hiệu IRi được đưa qua mạch khuếch đại đệm để tạo ra số hiệu ngắt tương ứng, số hiệu ngắt này sẽ được CPU đọc vào khi nó đưa ra tín hiệu trả lời /INTA. Bảng 6.1 cho ta quan hệ giữa IRi và số hiệu ngắt n tương ứng.

Lưu ý rằng Intel đã quy định INT 2 là số hiệu ngắt cứng do tín hiệu tích cực tại chân NMI gây ra.

Các số hiệu ngắt n trong INT n đều tương ứng với các địa chỉ xác định của chương trình con phục vụ ngắt chứa trong bảng các vectơ ngắt. Intel quy định bảng này nằm trong bộ nhớ trung tâm bắt đầu từ địa chỉ 00000H và có dung lượng cực đại là 1 hoặc 2KB (tuỳ theo cách khai báo vì bộ vi xử lý 16/32 bit có tất cả 256 ngắt, mỗi ngắt ứng với 1 vectơ ngắt, mỗi vectơ ngắt cần tối thiểu 4 byte để chứa địa chỉ đầy đủ cho CS: IP của chương trình con phục vụ ngắt).

Bảng vectơ ngắt được tổ chức theo cấu trúc sau (Bảng 6.2)

Bảng 6.2

03FEH- 03FFH	chứa giá trị con trỏ CS của CTCPVN FFH
03FCH- 03FDH	chứa giá trị con trỏ IP của CTCPVN FFH

0082H- 0083H	chứa giá trị con trỏ CS của CTCPVN 20H
0080H- 081H	chứa giá trị con trỏ IP của CTCPVN 20H

000AH- 000BH	chứa giá trị con trỏ CS của CTCPVN 2
0080H- 0009H	chứa giá trị con trỏ IP của CTCPVN 2
0006H- 0007H	chứa giá trị con trỏ CS của CTCPVN 1
0004H- 0005H	chứa giá trị con trỏ IP của CTCPVN 1
0002H- 0003H	chứa giá trị con trỏ CS của CTCPVN 0
0000H- 0001H	chứa giá trị con trỏ IP của CTCPVN 0

Chú thích: CTCPVN – chương trình con phục vụ ngắt

6.3. CHIP ĐIỀU KHIỂN NGẮT ƯU TIÊN PIC 8259A

6.3.1. Khái niệm ngắt ưu tiên

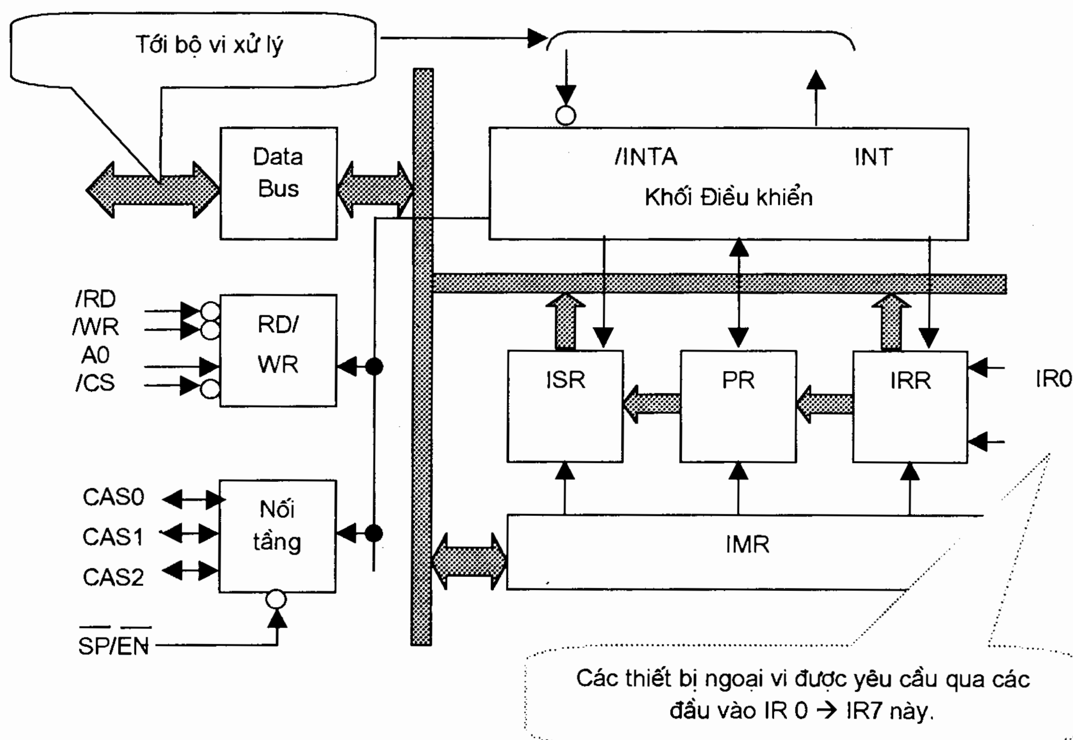
Nếu tại một thời điểm cùng có nhiều yêu cầu ngắt khác nhau cùng đòi hỏi bộ vi xử lý phục vụ thì bộ vi xử lý xử lý các yêu cầu ngắt đó như thế nào? Trong trường hợp như vậy, bộ vi xử lý sẽ xử lý các yêu cầu ngắt đó theo thứ tự ưu tiên, tức là ngắt nào có mức ưu tiên cao nhất hiện hành sẽ được bộ vi xử lý phục vụ trước.

Ngay trong cấu trúc phần cứng của bộ vi xử lý đã được quy định mức ưu tiên khác nhau cho các đầu ngắt như đầu ngắt NMI có mức ưu tiên cao hơn đầu ngắt INTR.

Giả sử tại một thời điểm nào đó bộ vi xử lý cùng nhận được yêu cầu ngắt từ đầu vào INTR và đầu vào NMI. Bộ vi xử lý sẽ xử lý ra sao trong trường hợp này?

Theo thứ tự ưu tiên ngầm định trong việc xử lý ngắt của bộ vi xử lý thì NMI có mức ưu tiên cao hơn INTR, vì vậy đầu tiên bộ vi xử lý sẽ thực hiện chương trình phục vụ ngắt INT 2 (là ngắt NMI) để xử lý yêu cầu của ngắt này. Lúc này cờ IF sẽ tự động bị xoá về 0. Yêu cầu ngắt trên đầu INTR bị cấm cho tới khi chương trình phục vụ ngắt NMI hoàn tất công việc của mình và trở về bằng lệnh IRET thì cờ IF được khôi phục. Lúc này bộ vi xử lý mới kích hoạt chương trình con dành cho đầu ngắt ITR.

6.3.2. Chip điều khiển ngắt ưu tiên 8259A



Hình 6.6. **Sơ đồ cấu trúc của chip ngắt ưu tiên PIC 8259**

Khi hệ vi xử lý có nhiều ngoại vi được ghép nối và làm việc theo cơ chế ngắt thì phải mở rộng các đầu ngắt để đáp ứng về số lượng ngắt. Khi số lượng ngắt tăng lên thì vấn đề quy định mức ưu tiên phải được đặt ra vì bộ vi xử lý tại một thời điểm chỉ có thể phục vụ cho một đầu ngắt mà thôi.

Trong trường hợp như vậy, giải pháp đúng đắn hơn cả là sử dụng chip IC chuyên dụng cho phép mở rộng bảng vectơ ngắt cứng lên số lượng cần thiết. Chip 8259A được gọi là mạch điều khiển ngắt ưu tiên - PIC (priority interrupt controller). Đó là một vi mạch cỡ lớn lập trình được. Nó có thể xử lý được 8 yêu cầu ngắt với 8 mức ưu tiên khác nhau để tạo ra một yêu cầu ngắt đưa đến đầu vào chung là INTR (yêu cầu ngắt che được) của bộ vi xử lý. Nếu nối tầng một mạch 8259A chủ với 8 mạch 8259A thợ ta có thể nâng tổng số các yêu cầu ngắt với các mức ưu tiên khác nhau lên thành 64 đầu vào ngắt độc lập.

Sơ đồ cấu trúc của chip 8259 được thể hiện trên hình 6.6.

Chú thích:

IRO-IR7 (Interrupt Requests): Các đầu tín hiệu yêu cầu ngắt.

IRR (Interrupt Request Register): Thanh ghi chứa yêu cầu ngắt.

PR (Priority Resolve): Giải quyết mức ngắt ưu tiên.

SP/EN (Slave program/Enable buffer) lập trình cho mạch PIC 8259 thợ/ mở đệm bus dữ liệu.

ISR (In Service Register): Thanh ghi chứa yêu cầu ngắt đang được phục vụ.

Cas0 - Cas2 (Cascades): Tín hiệu nối tầng giữa các PIC 8259 với nhau.

Các khối chức năng chính của 8259A bao gồm:

- Thanh ghi IRR: Ghi nhớ các yêu cầu ngắt có tại đầu vào IRi. Bất cứ ngoại vi nào có yêu cầu ngắt (trường hợp cực đại là cả 8 đầu vào đều có yêu cầu) sẽ được thanh ghi này ghi nhận.
- Thanh ghi ISR: ghi nhớ yêu cầu ngắt đang được phục vụ trong số các yêu cầu ngắt IRi đang có trong mạch.
- Thanh ghi IMR: Ghi nhớ mặt nạ ngắt đối với các yêu cầu ngắt IRi.
- Logic điều khiển : khối này có nhiệm vụ gửi yêu cầu ngắt tới bộ vi xử lý qua INTR khi có tín hiệu tại các chân IRi và nhận tín hiệu chấp nhận yêu cầu ngắt INTA từ phía CPU để rồi điều khiển việc đưa ra kênh dữ liệu số hiệu ngắt.
- Đệm kênh dữ liệu: dùng để phối ghép 8259A với kênh dữ liệu của CPU.
- Logic điều khiển RD/WR: dùng cho việc ghi các từ điều khiển và đọc các từ trạng thái của 8259A.

- Khối đệm nối tầng và so sánh: ghi nhớ và so sánh số hiệu của các mạch 8259A có mặt trong hệ vi xử lý.

Một số tín hiệu trong mạch 8259 có tên giống như các tín hiệu tiêu chuẩn của hệ vi xử lý 80x86. Ngoài các tín hiệu này ra còn có một số tín hiệu đặc biệt khác của 8259A cần phải giới thiệu thêm gồm:

- Cas_0 - Cas_2 [I,O] : là các đầu vào đối với các mạch 8259A thợ hoặc các đầu ra của mạch 8259A và chỉ dùng khi cần nối tầng để tăng thêm số lượng đầu vào yêu cầu ngắt cần xử lý.
- SP/EN [I,O]: Khi 8259A làm việc ở chế độ không có đệm bus dữ liệu thì đây là tín hiệu vào dùng để lập trình để biến mạch 8259A thành mạch thợ (SP=0) hoặc chủ (SP=1). Khi 8259A làm việc trong hệ vi xử lý ở chế độ có đệm bus dữ liệu thì chân này là tín hiệu ra EN dùng mở đệm bus dữ liệu để bộ vi xử lý và 8259A thông vào bus dữ liệu hệ thống. Lúc này việc định nghĩa mạch 8259A là chủ hoặc thợ phải thực hiện thông qua từ điều khiển khởi đầu ICW4.
- INT [O] : Tín hiệu yêu cầu ngắt đến chân INTR của bộ vi xử lý.
- INTA [I] : Nối với tín hiệu báo chấp nhận ngắt INTA của bộ vi xử lý.

6.3.2. Lập chế độ làm việc cho chip 8259A

Để mạch PIC PIC 8259A có thể hoạt động được theo yêu cầu. Sau khi bật nguồn cấp điện, PIC cần phải được lập trình bằng cách ghi vào thanh ghi (tương đương với các cổng IO) bên trong nó các từ điều khiển khởi đầu (ICW) và tiếp sau đó là các từ điều khiển hoạt động (OCW).

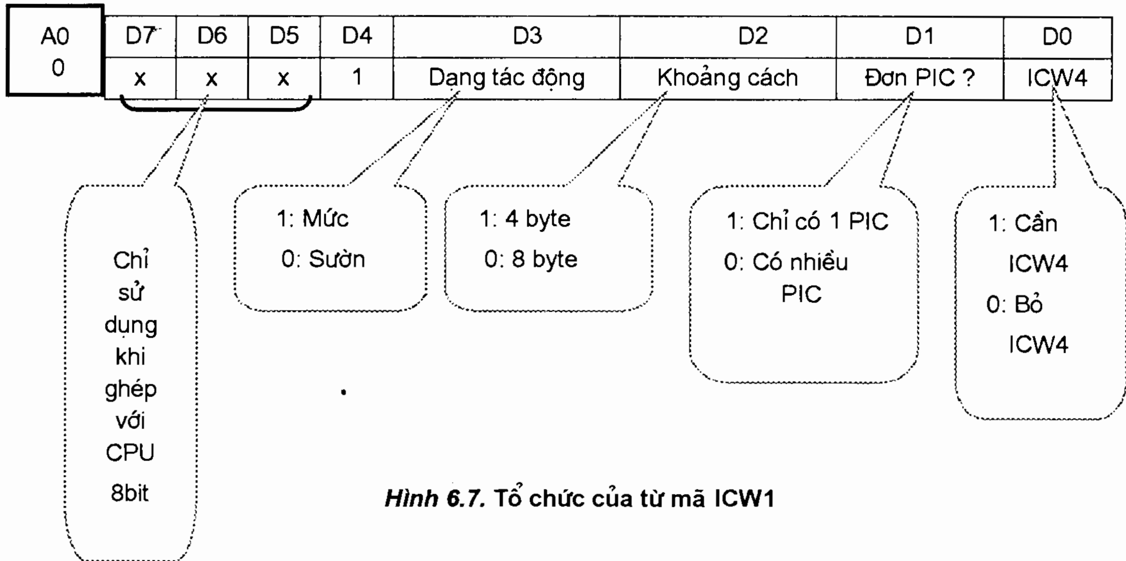
Các từ điều khiển khởi đầu dùng để tạo nên các chế độ (Mode) làm việc cơ bản cho PIC còn các từ điều khiển hoạt động sẽ quyết định cách thức làm việc cụ thể của PIC. Từ điều khiển hoạt động sẽ được ghi khi ta muốn thay đổi hoạt động của PIC.

Các từ điều khiển khởi đầu ICW

PIC 8259A có tất cả 4 từ điều khiển khởi đầu là ICW1, ICW2, ICW3 và ICW4. Trong khi lập trình cho PIC không phải lúc nào ta cũng cần dùng cả 4 từ điều khiển đó. Tuỳ theo các trường hợp ứng dụng cụ thể mà có lúc cần ghi liên tiếp cả 4 từ điều khiển khởi đầu nhưng có lúc chỉ cần ghi vào đó 2 hay 3 từ là đủ.

Từ mã khởi đầu ICW 1

Tổ chức của từ mã khởi đầu ICW 1 (Initial Control Word 1) có dạng như hình 6.7.



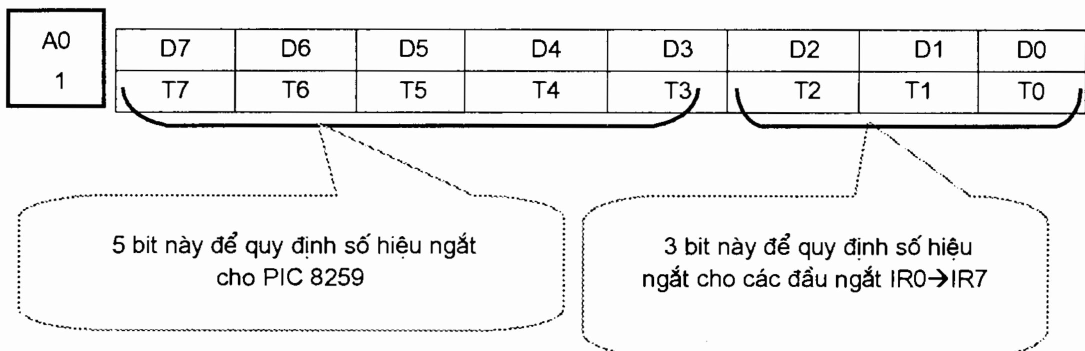
Hình 6.7. **Tổ chức của từ mã ICW1**

Bit D0 của ICW1 quy định 8259A sẽ được nối với họ vi xử lý nào. Để làm việc với hệ 16/32 bit (họ 80x86) thì ICW1 nhất thiết phải có IC4 = 1 (tức là ta luôn cần đến ICW4) còn đối với hệ 8 bit như họ 8080/85 thì phải có IC4 = 0 (và như vậy các bit của ICW4 sẽ buộc bị xoá về 0). Các bit còn lại của ICW1 định nghĩa cách thức tác động của xung yêu cầu ngắt (tác động theo sườn hay theo mức) tại các chân yêu cầu ngắt IR của mạch 8259A khác trong hệ làm việc đơn lẻ hay theo chế độ nối tầng. Bit D2 = 1 thì mỗi vector ngắt được cấp phát 4 byte còn D2 = 0 thì mỗi vector ngắt được cấp phát 8 byte.

Các bit được đánh dấu x là không quan trọng và thường được lấy giá trị 0 để lập trình cho các ứng dụng sau này.

Từ mã khởi đầu ICW 2

Tổ chức của từ mã khởi đầu ICW 2 (Initial Control Word 2) có dạng như hình 6.8.



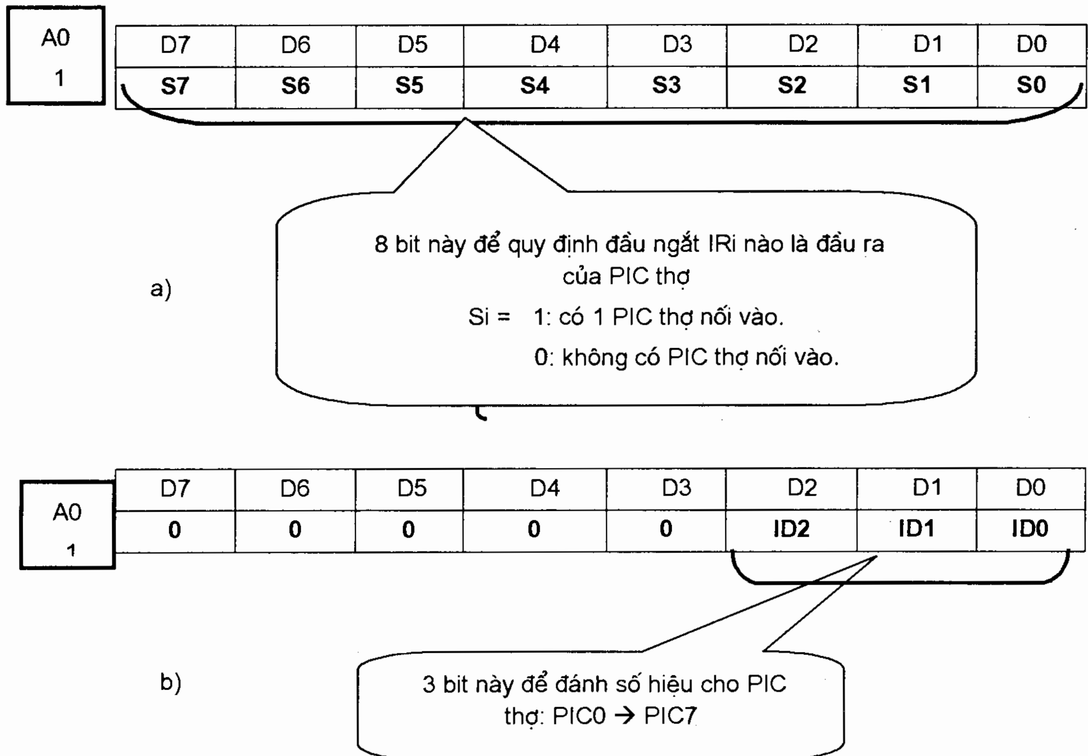
Hình 6.8. **Tổ chức của từ mã ICW2**

Từ điều khiển khởi đầu này cho phép người thiết kế lựa chọn số hiệu ngắt cho PIC 8259 ứng bằng các bit T3-T7. Các bit T0-T2 được 8259A tự động gán giá trị tuỳ theo đầu vào yêu cầu ngắt cụ thể IRi. Ví dụ nếu ta muốn các đầu vào của mạch 8259A có số hiệu ngắt là 40-47H ta chỉ cần ghi giá trị 40H vào các bit T3-T7. Nếu làm như vậy thì IR0 sẽ có số hiệu ngắt là 40H, IR1 sẽ có số hiệu ngắt là 41H... IR7 sẽ có số hiệu ngắt là 47H.

Từ mã khởi đầu ICW 3

Từ điều khiển khởi đầu này chỉ dùng đến khi bit D1 (Đơn PIC ?) thuộc từ điều khiển khởi đầu ICW1 có giá trị 0 nghĩa là trong hệ có các mạch 8259A làm việc ở chế độ nối tầng. Chính vì vậy tồn tại hai loại ICW3:một cho mạch 8259A chủ và một cho mạch 8259A thợ.

Tổ chức của từ mã khởi đầu ICW 3 (Initial Control Word 3) có dạng như hình 6.9.



Hình 6.9. **Tổ chức của từ mã ICW3: a) cho PIC chủ, b) cho PIC thợ**

ICW3 cho mạch chủ dùng để chỉ ra đầu vào yêu cầu ngắt IRi nào của nó có tín hiệu INT của mạch thợ nối vào.

ICW3 cho mạch thợ dùng làm phương tiện để các mạch này được nhận biết. Vì vậy từ điều khiển khởi đầu ICW3 này phải chứa mã số ***i*** ứng với đầu vào *IRi* của mạch chủ mà mạch thợ đã được nối vào. Mạch thợ sẽ so sánh mã số này

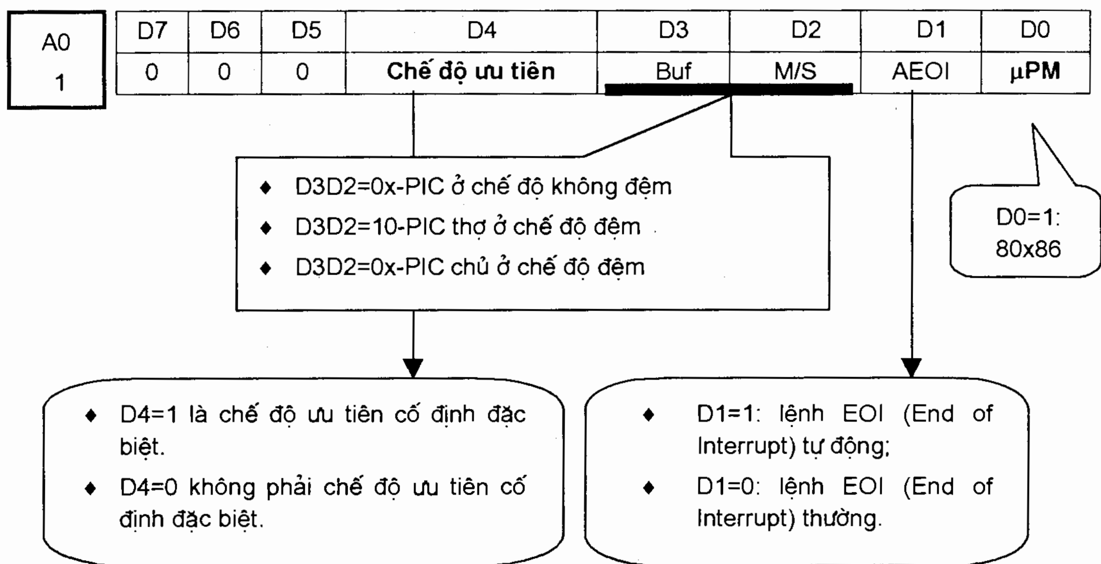
với mã số nhận được ở 3 bit nối tầng CaS2 CaS1 Cas0, nếu bằng nhau thì số hiệu ngắt của PIC thợ sẽ được đưa ra kênh dữ liệu khi có xung /INTA. Ví dụ:

Trong một hệ vi xử lý ta có một mạch 8259A chủ và hai mạch 8259A thợ nối vào chân IR0 và IR1 của mạch chủ, còn các chân IR khác của PIC chủ được nối trực tiếp với các ngoại vi của hệ. Tìm giá trị phải gán cho các từ điều khiển khởi đầu ICW3.

Như vậy để tổ hợp các PIC này làm việc được với nhau ta sẽ phải ghi các từ điều khiển khởi đầu như sau: ICW3 = 03H cho mạch chủ, ICW3 = 00H cho mạch thợ thứ nhất và ICW3 = 01H cho mạch thợ thứ hai.

Từ mã khởi đầu ICW 4

Tổ chức của từ mã khởi đầu ICW 4 (Initial Control Word 4) có dạng như hình 6.10.

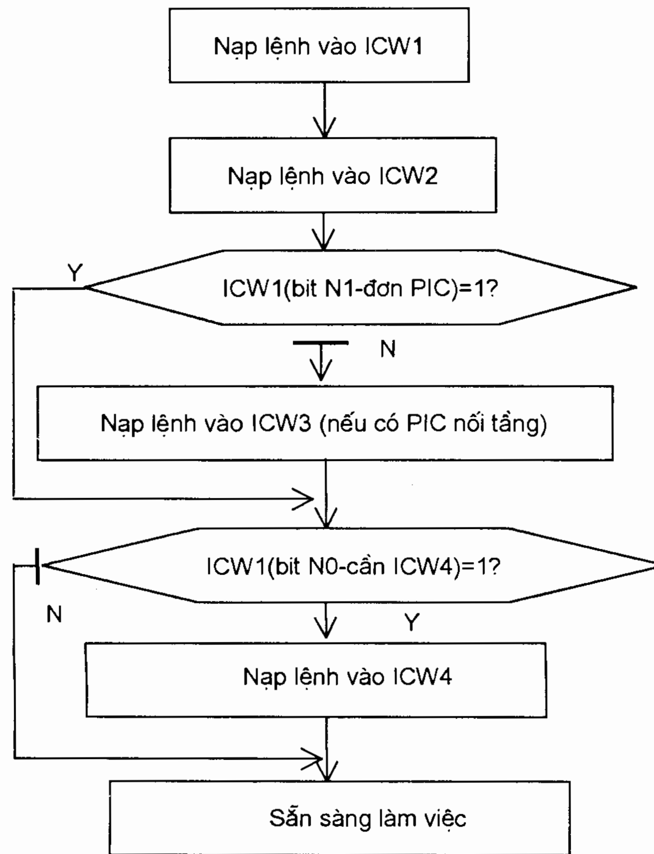


Hình 6.10. **Tổ chức của từ mã ICW4**

Từ điều khiển khởi đầu này chỉ dùng đến khi trong từ điều khiển ICW1 có D0 = 1 (cần thêm ICW4).

Bit μPM (microprocessor mode) cho ta khả năng chọn loại vi xử lý để làm việc với 8259A. Bit μPM =1 cho phép các bộ vi xử lý từ 80x86 làm việc với 8259A .

Bit *Chế độ ưu tiên* = 1 cho phép chọn chế độ ưu tiên cố định đặc biệt. Trong chế độ này yêu cầu ngắt với mức ưu tiên cao nhất hiện hành từ một mạch thợ làm việc theo kiểu nối tầng sẽ được mạch chủ nhận biết ngay cả khi mạch chủ còn đang phải phục vụ một yêu cầu ngắt ở mạch thợ khác nhưng với mức ưu tiên thấp hơn. Sau khi các yêu cầu ngắt được phục vụ xong thì chương trình phục vụ ngắt phải có lệnh kết thúc yêu cầu ngắt EOI (End Of Interrupt) đặt trước lệnh trở về IRET đưa đến cho mạch 8259A chủ.



***Hình 6.11.* Trình tự khởi động PIC 8259**

Khi bit *Chế độ ưu tiên* SFNM (SPECIAL FULLY NESTED MODE) = 0 thì chế độ ưu tiên cố định được chọn (IRO là mức ưu tiên cao nhất còn IR7 là mức ưu tiên thấp nhất). Thực ra đối với mạch 8259A không dùng đến ICW4 thì chế độ này đã được chọn như là ngầm định. Trong chế độ ưu tiến cố định tại một thời điểm chỉ có một yêu cầu ngắt i được phục vụ (bit ISRi =1) lúc này tất cả các yêu cầu khác với mức ưu tiên thấp hơn đều bị cấm. Tất cả các yêu cầu khác với mức ưu tiên cao hơn có thể ngắt yêu cầu khác với mức ưu tiên thấp hơn.

Bit BUF (BUFFER) cho phép định nghĩa mạch 8259A để làm việc với CPU trong trường hợp có đệm hoặc không có đệm nối với kênh hệ thống. Khi làm việc ở chế độ có đệm (BUF=1) bit M/S =1/0 cho phép ta chọn mạch 8259A để làm việc ở chế độ chủ / thợ. SP/EN trở thành đầu ra cho phép mở đệm để PIC và CPU thông với kênh hệ thống.

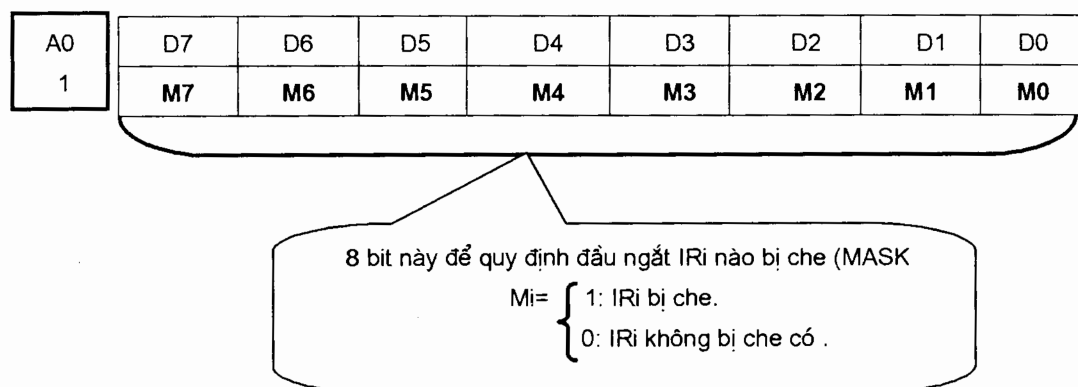
Lưu ý rằng bên cạnh các bit dữ liệu của từ điều khiển khởi đầu ICW cần ghi rõ cả giá trị của bit A0 tương ứng cho mỗi ICW. Đầu vào địa chỉ A0 và thứ tự ghi sẽ giúp phân biệt ra các thanh ghi khác nhau nằm bên trong PIC để ghi dữ liệu cho các từ điều khiển. Ví dụ A0 = 0 là dấu hiệu để nhận biết rằng ICW1

được đưa vào thanh ghi có địa chỉ chẵn trong PIC. Còn khi A0 = 1 thì các từ điều khiển khởi đầu ICW2, ICW3, ICW4 sẽ được đưa vào các thanh ghi có địa chỉ lẻ trong mạch PIC.

Các từ điều khiển hoạt động OCW

Các từ điều khiển hoạt động OCW (Operation Control Worrd) sẽ quy định chế độ hoạt động của PIC 8259A sau khi nó đã được khởi đầu bằng các từ điều khiển ICW. Tất cả các từ điều khiển hoạt động sẽ được ghi vào các thanh ghi trong PIC khi bit A0 = 0, trừ OCW1 được ghi khi A0 = 1.

Từ mã điều khiển hoạt động OCW1



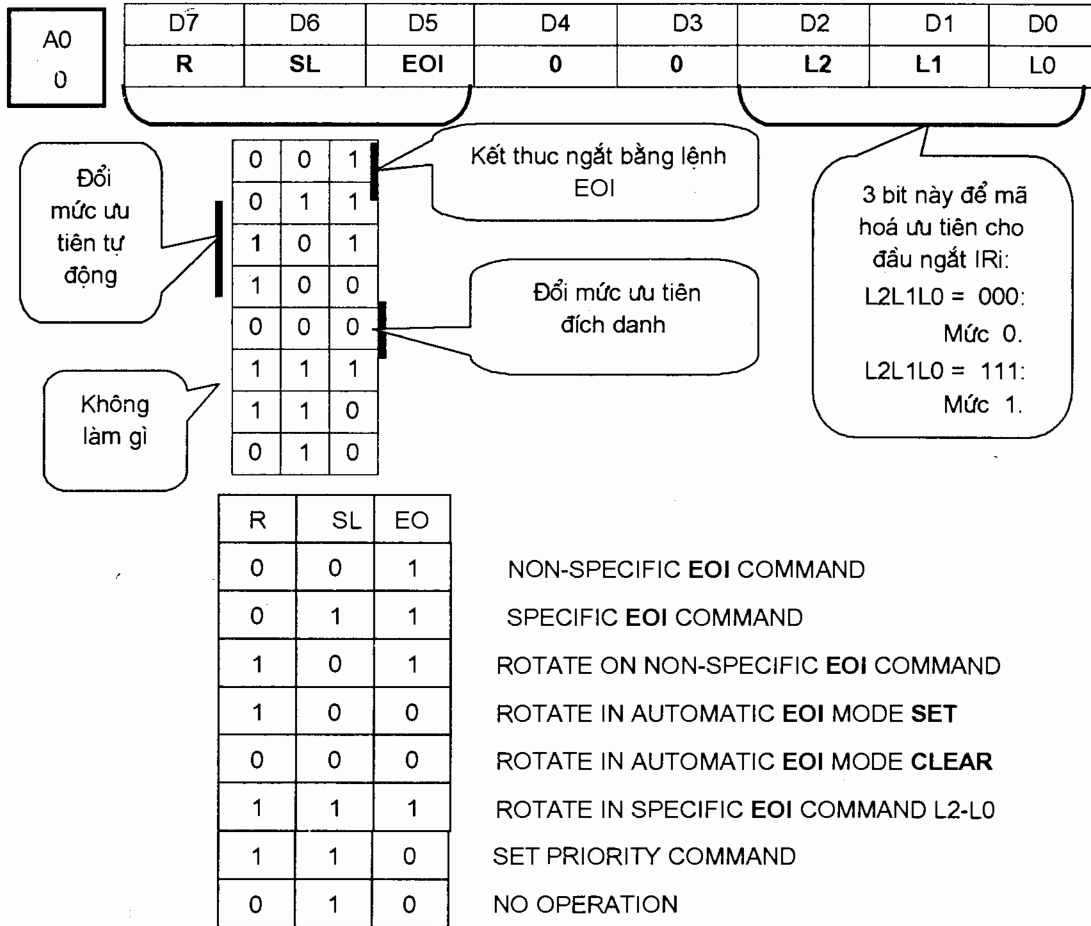
Hình 6.12. **Tổ chức của từ mã OCW1**

OCW1 dùng để ghi giá trị của các bit mặt nạ vào thanh ghi mặt nạ ngắt IMR. Khi một bit mặt nạ nào đó của IMR được lập thì yêu cầu ngắt tương ứng với mặt nạ đó sẽ không được 8259A nhận biết nữa (nó đã bị che). Từ điều khiển này phải được đưa đến 8259A ngay sau khi ghi các ICW vào 8259A.

Ta cũng có thể đọc lại IMR để xác định tình trạng mặt nạ ngắt hiện tại để kiểm tra xem trong thời điểm hiện tại yêu cầu ngắt nào bị che. Tổ chức của từ mã lệnh này được thể hiện trên hình 6.12.

Từ mã điều khiển hoạt động OCW2

Từ mã điều khiển hoạt động OCW2 là từ mã điều khiển cho phép đặt mức ưu tiên rất mềm dẻo cho các đầu vào nối với thiết bị ngoại vi. Tổ chức của từ mã lệnh này được thể hiện trên hình 6.13.



R	SL	EO	
0	0	1	NON-SPECIFIC **EOI** COMMAND
0	1	1	SPECIFIC **EOI** COMMAND
1	0	1	ROTATE ON NON-SPECIFIC **EOI** COMMAND
1	0	0	ROTATE IN AUTOMATIC **EOI** MODE **SET**
0	0	0	ROTATE IN AUTOMATIC **EOI** MODE **CLEAR**
1	1	1	ROTATE IN SPECIFIC **EOI** COMMAND L2-L0
1	1	0	SET PRIORITY COMMAND
0	1	0	NO OPERATION

Hình 6.13. **Tổ chức của từ mã OCW2**

Các bit R (*ROTATE*), SL (*SPECIFIC*) và EOI (*END OF INTERRUPT*) phối hợp với nhau cho phép chọn ra các cách thức kết thúc yêu cầu ngắt khác nhau. Một vài cách kết thúc yêu cầu ngắt còn tác động tới các yêu cầu ngắt khác nhau. Một vài cách kết thúc yếu cầu ngắt còn tác động tới các yêu cầu ngắt được chỉ đích danh với mức ưu tiên được mã hoá bởi 3 bit L_2,L_1,L_0. da Chúng ta phân tích chức năng điều khiển của từng tổ hợp giá trị của *R, SL và EOI*

R SL EOI = 001 là mã lệnh EOI thông thường.

R SL EOI = 011 là mã lệnh EOI chỉ đích danh ngắt có số hiệu là giá trị các bit L_2, L_1, L_0.

R SL EOI = 101 là mã lệnh đổi mức ưu tiên khi có EOI thường.

R SL EOI = 100 là mã lệnh lập chế độ quay khi có EOI tự động.

R SL EOI = 000 là mã lệnh xoá chế độ quay khi có EOI tự động.

R SL EOI = 111 là mã lệnh đổi mức ưu tiên khi có EOI chỉ đích danh ngắt có số hiệu là giá trị các bit L_2, L_1, L_0.

R SL EOI = 110 là mã lệnh lập mức ưu tiên cho ngắt có số hiệu là giá trị các bit L_2, L_1, L_0.

Trước khi nói về các cách kết thúc yêu cầu ngắt cho các chế độ ta cần xét các chế độ làm việc của 8259A.

♦ *Chế độ ưu tiên cố định:*

Đây là chế độ làm việc ngầm định của 8259A sau khi nó đã được nạp các từ điều khiển khởi đầu để thiết lập trạng thái xác định. Trong chế độ này, các đầu vào IR7-IR0 được gán cho các mức ưu tiên cố định. Đầu vào IRO được gán cho mức ưu tiên cao nhất, kế đến là IR1, cuối cùng là IR7 có mức ưu tiên thấp nhất. Mức ưu tiên này được giữ không thay đổi cho đến khi chip PIC 8259A bị lập trình khác đi bằng OCW2.

Trong chế độ ưu tiên cố định tại một thời điểm chỉ có một yêu cầu ngắt i được phục vụ (bit ISRi = 1). Lúc này tất cả các yêu cầu khác với mức ưu tiên thấp hơn đều bị cấm, tất cả các yêu cầu khác với mức ưu tiên cao hơn có thể ngắt yêu cầu khác với mức ưu tiên thấp hơn.

♦ *Chế độ quay mức ưu tiên tự động:*

Trong chế độ này sau khi một yêu cầu ngắt được phục vụ xong, PIC 8259A sẽ xoá bit tương ứng của nó trong thanh ghi ISR và gán cho đầu vào này mức ưu tiên thấp nhất để tạo điều kiện cho các yêu cầu ngắt khác có thời cơ được phục vụ.

♦ *Chế độ quay mức ưu tiên chỉ đích danh:*

Trong chế độ này cần chỉ đích danh đầu vào IRi nào với $i = L_2L_1L_0$ được gán mức ưu tiên thấp nhất còn đầu vào IR_{i+1} sẽ được tự động gán mức ưu tiên cao nhất.

Các bit R, SL và EOI phải phối hợp với nhau để tạo ra các lệnh quy định các cách thức kết thúc yêu cầu ngắt cho các chế độ làm việc khác nhau .

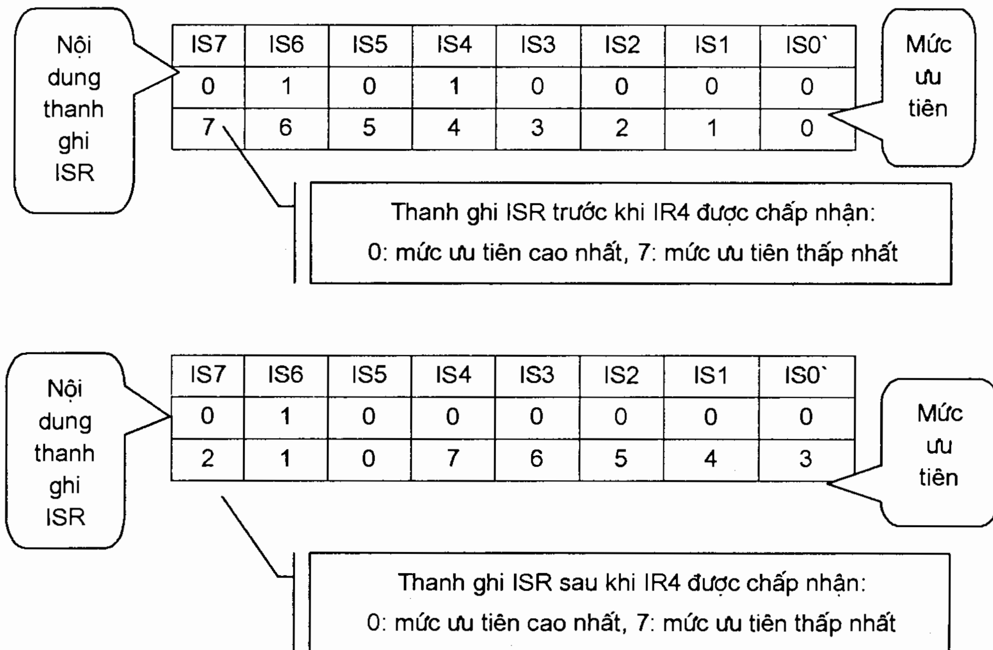
♦ *Kết thúc yêu cầu ngắt thường:*

Chương trình con phục vụ ngắt phải có lệnh EOI đặt trước lệnh trở về IRET cho 8259A. Nghĩa là mỗi khi kết thúc chương trình con phục vụ ngắt, trước khi đưa lệnh IRET cho CPU thì phải đưa mã lệnh EOI (=20h) cho PIC 8259A. Mạch 8259A sẽ xác định yêu cầu ngắt IRi vừa được phục vụ và xoá bit ISRi tương ứng của nó để tạo điều kiện cho chính yêu cầu ngắt này hoặc các ngắt khác có mức ưu tiên thấp hơn có thể được tác động.

Kết thúc yêu cầu ngắt chỉ đích danh: Chương trình con phục vụ ngắt phải có lệnh EOI chỉ đích danh đặt trước lệnh trở về IRET cho 8259A . Mạch 8259A xoá đích danh bit ISRi với $i = L_2L_1L_0$ để tạo điều kiện cho chính yêu cầu ngắt này hoặc các ngắt khác có mức ưu tiên thấp hơn có thể được tác động.

Quay vòng mức ưu tiên khi kết thúc *yêu cầu ngắt thường* thì chương trình con phục vụ ngắt phải có lệnh EOI đặt trước lệnh trở về IRET cho 8259A. Mạch 8259A sẽ xác định yêu cầu ngắt thứ i vừa được phục vụ và xoá bit ISRi tương ứng rồi gán luôn mức ưu tiên thấp nhất cho đầu vào IR_i này, còn đầu vào IR_{i+1} sẽ được gán mức ưu tiên cao nhất.

Có thể theo dõi cách thức hoạt động của mạch 8259A trong chế độ quay vòng mức ưu tiên khi kết thúc yêu cầu ngắt thường thông qua ví dụ minh hoạ trình bày trên hình 6.14.



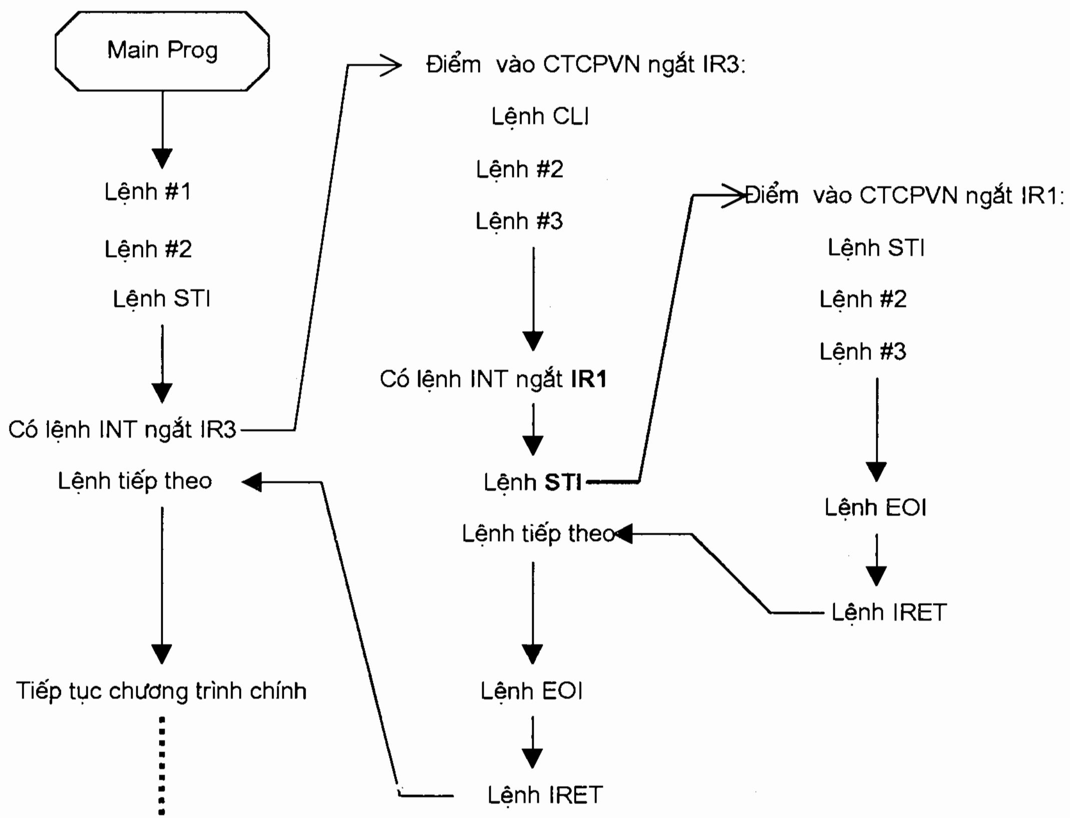
Hình 6.14. **Quay vòng mức ưu tiên khi kết thúc yêu cầu ngắt thường**

- Quay vòng mức ưu tiên trong chế độ kết thúc yêu cầu ngắt tự động thì chỉ cần một lần đưa lệnh chọn chế độ đổi mức ưu tiên khi kết thúc yêu cầu ngắt tự động. Có thể chọn chế độ này bằng lệnh *lập chế độ quay khi có EOI tự động*. Từ đó trở đi, 8259A sẽ đổi mức ưu tiên mỗi khi kết thúc ngắt tự động theo cách tương tự như trên. Muốn bỏ chế độ này ta có thể dùng lệnh xoá *chế độ quay khi có EOI tự động*.

- Quay vòng mức ưu tiên khi kết thúc yêu cầu ngắt đích danh thì chương trình con phục vụ ngắt phải có lệnh EOI đích danh cho 8259A đặt trước lệnh trở về IRET. Mạch 8259A sẽ xoá bit ISRi của yêu cầu ngắt tương ứng và gán luôn mức ưu tiên thấp nhất cho đầu vào IRi với $i = L_2L_1L_0$. Yêu cầu ngắt IR_{1+1} sẽ được gán mức ưu tiên cao nhất.

- Lập mức ưu tiên: Chế độ này cho phép thay đổi mức ưu tiên cố định hoặc mức ưu tiên gán trước đó bằng cách gán mức ưu tiên thấp nhất cho yêu cầu ngắt IRi chỉ đích danh ứng với tổ hợp mã **i**=$L_2L_1L_0$. Yêu cầu ngắt IR $_{1+1}$ sẽ được gán mức ưu tiên cao nhất.

Thí dụ minh hoạ cho chế độ làm việc thông dụng nhất là chế độ ưu tiên cố định:

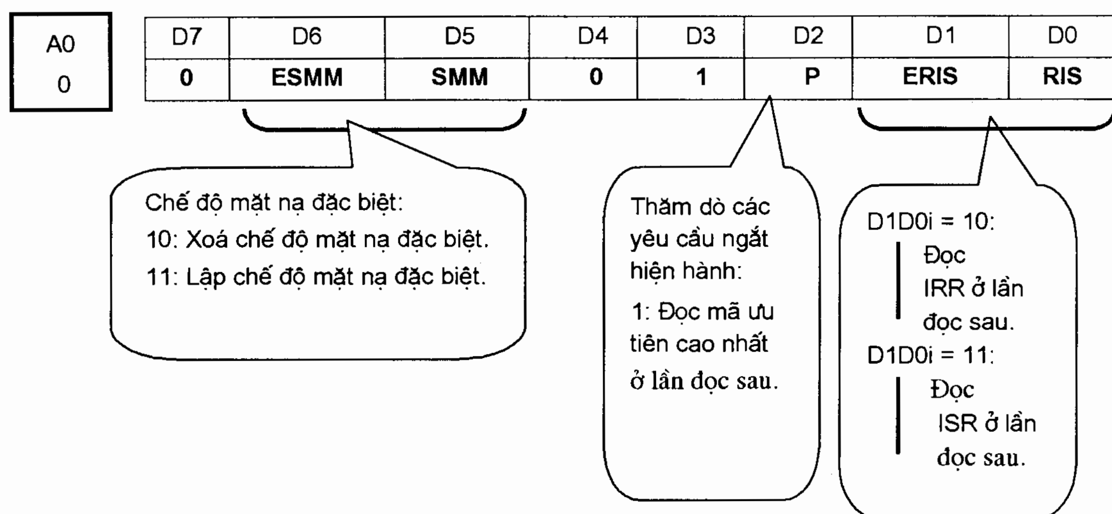


Mô tả hoạt động của chương trình minh hoạ: Trong quá trình thực hiện chương trình chính, IR3 có yêu cầu ngắt. Do được phép ngắt, CPU chuyển điều khiển tới chương trình con phục vụ ngắt IR3. Trong khi thực hiện chương trình con phục vụ ngắt IR3 thì ở đầu vào IR1 xuất hiện yêu cầu ngắt. Vì IR1 có mức ưu tiên cao hơn IR3 nên PIC 8259 phải tạo ngắt cho IR1. Song do khi ghi nhận ngắt IR3, CPU bị cấm ngắt, nên IR1 chỉ được ghi nhận khi CPU thực hiện lệnh STI có trong chương trình con phục vụ ngắt IR3. IR3 cần bảo vệ khỏi các yêu cầu ngắt khác, còn IR1 không cần nên lệnh STI được đặt ngay ở đầu chương trình con. Trong quá trình hoạt động cả hai bit ISR1 và ISR3 đều được đặt = 1, nghĩa là cả hai chương trình con phục vụ ngắt IR1 và IR3 đều chưa hoàn thành.

Trong giai đoạn này chỉ có IR0 được quyền ngắt vì nó là mức ưu tiên cao nhất. Để kết thúc chương trình con phục vụ ngắt IR3, ở cuối chương trình con này phải có lệnh EOI, PIC căn cứ vào đó sẽ xoá bit ISR1. Còn với lệnh IRET ở cuối chương trình con, CPU chuyển điều khiển lại cho chương trình con phục vụ ngắt IR3. Lúc này cả ba IR0, IR1, IR2 đều có thể ngắt PIC vì trong ISR chỉ còn bit ISR3 = 1..Theo thí dụ này, vì không còn nguồn gây ngắt tiếp nên cuối chương trình con IR3 lệnh EOI sẽ xoá bit ISR3 và CPU quay lại chương trình chính với lệnh IRET.

Từ mã điều khiển hoạt động OCW3

Tổ chức của từ mã lệnh này được thể hiện trên hình 6.15.

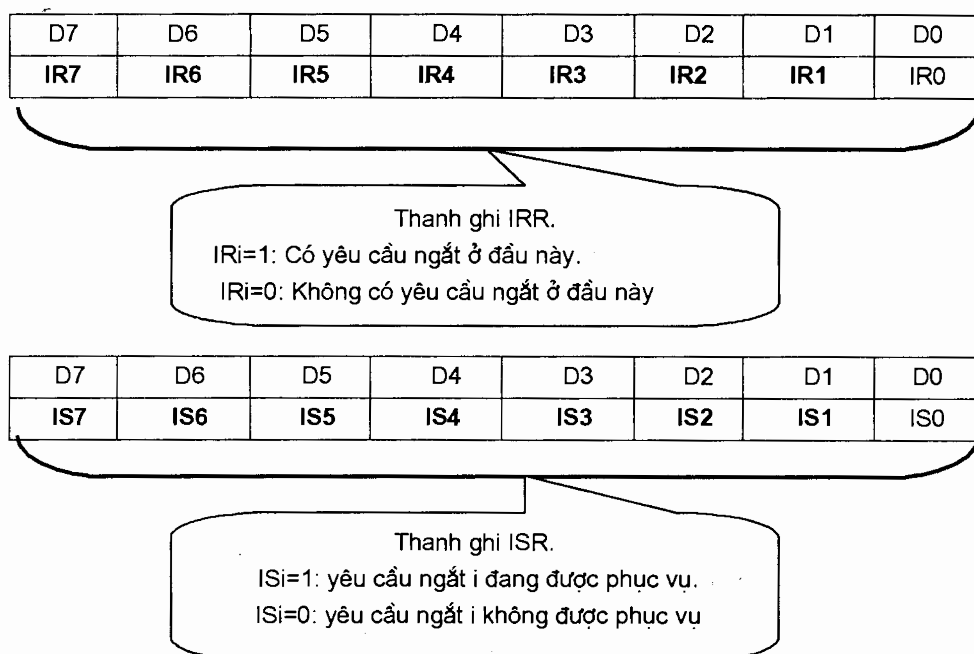


Hình 6.15. **Tổ chức của từ mã OCW3**

Từ điều khiển hoạt động OCW3 sau khi được ghi vào 8259A cho phép:

- Chọn ra các thanh ghi để đọc
- Thăm dò trạng thái yêu cầu ngắt bằng cách đọc trạng thái của đầu vào yêu cầu ngắt IRi với mức ưu tiên cao nhất.
- Thao tác với mặt nạ đặc biệt.

Các thanh ghi IRR và ISR có thể đọc được sau khi nạp vào 8259A từ điều khiển OCW3 với tổ hợp bit ERIS-RIS (Enable Read IS Register-Read IS Register) =10 sẽ cho phép đọc IRR còn tổ hợp bit ERIS-RIS = 11 sẽ cho phép đọc ISR. Tổ chức của các thanh ghi này được biểu diễn trên hình 6.16.



***Hình 6.16.* Tổ chức của thanh ghi IRR và ISR**

Bằng việc đưa vào PIC 8259A từ điều khiển OCW3 với bit P (POLLING COMMAND) = 1 ta có thể đọc được trên kênh dữ liệu ở lần đọc tiếp ngay sau đó từ mã thăm dò, trong đó có các thông tin về yêu cầu ngắt với mức ưu tiên cao nhất đang hoạt động và từ mã tương ứng với yêu cầu ngắt ấy.

Có thể gọi đây là chế độ thăm dò yêu cầu ngắt và chế độ này thường được ứng dụng trong trường hợp có nhiều chương trình phục vụ ngắt giống nhau cho một yêu cầu ngắt và việc chọn chương trình nào để sử dụng là trách nhiệm của người lập trình.

Như vậy, muốn dùng chế độ thăm dò của 8259A để xác định yêu cầu ngắt hiện tại ta cần làm các thao tác lần lượt như sau:

- Cấm các yêu cầu ngắt bằng lệnh CLI.
- Ghi từ mã lệnh OCW3 có bit P=1.
- Đọc từ thăm dò trạng thái yêu cầu ngắt trên kênh dữ liệu.

Bit ESMM (Enable Special Mask Mode)=1 cho phép 8259A thao tác với chế độ mặt nạ đặc biệt. Bit SMM(Special Mask Mode)=1 cho phép lập chế độ mặt nạ đặc biệt. Chế độ mặt nạ đặc biệt được dùng để thay đổi thứ tự ưu tiên ngay bên trong chương trình con phục vụ ngắt. Trong trường hợp có một yêu cầu ngắt bị cấm (bị che bởi chương trình phục vụ ngắt với từ lệnh OCW1) mà hệ vi xử lý lại muốn cho phép các yêu cầu ngắt với mức ưu tiên thấp hơn so với yêu cầu ngắt bị cấm đó được tác động thì ta sẽ dùng chế độ mặt nạ đặc biệt. Một khi

đã được lập, chế độ mặt nạ đặc biệt sẽ tồn tại cho tới khi bị xoá bằng cách ghi vào 8259A một từ lệnh OCW3 khác với bit SMM = 0. Mặt nạ đặc biệt không ảnh hưởng tới các yêu cầu ngắt với mức ưu tiên cao hơn.

Tóm lại, hoạt động của hệ vi xử lý với chế độ ngắt có sử dụng PIC 8259A để mở rộng số lượng vectơ ngắt cứng theo trình tự sau:

- Khi có yêu cầu ngắt từ thiết bị ngoại vi tác động vào một trong các chân IR của PIC 8259A thì PIC 8259A sẽ đưa tín hiệu INT=1 đến chân INTR của của bộ vi xử lý 80x86.
- Bộ vi xử lý đưa ra xung /INTA đầu đến đầu vào /INTA của PIC 8259A.
- PIC 8259A dùng xung /INTA thứ nhất (phát đi từ bộ vi xử lý) như là thông báo để nó hoàn tất các xử lý nội bộ cần thiết kể cả xử lý ưu tiên nếu như có nhiều yêu cầu ngắt cùng xảy ra.
- Bộ vi xử lý đưa ra xung /INTA thứ hai đến PIC 8259A.
- Xung /INTA thứ hai buộc PIC 8259A đưa ra kênh dữ liệu byte chứa thông tin về số hiệu ngắt của yêu cầu ngắt vừa được nhận biết.
- Bộ vi xử lý dùng số hiệu ngắt để tính ra địa chỉ của vectơ ngắt tương ứng bằng công thức: ***(số hiệu ngắt * 4).***
- Bộ vi xử lý lưu trữ thanh ghi cờ F vào ngăn xếp, xoá các cờ IF và TF và lưu trữ địa chỉ trở về CS:IP vào ngăn xếp.
- Bộ vi xử lý gán địa chỉ CS:IP của chương trình phục vụ ngắt từ bảng vectơ ngắt:

  (con trỏ CS:IP) ← ((số hiệu ngắt * 4))

  và thực hiện chương trình đó.

6.4. GHÉP NỐI CHIP 8259A VỚI HỆ VI XỬ LÝ

6.4.1. Sơ đồ Ghép nối Chip 8259A với hệ vi xử lý

Ghép PIC 8259A với hệ vi xử lý 80286 (mở rộng cho 80X86) được trình bầy trên hình 6.17. Các tín hiệu ghép nối phần lớn là tương thích nhau. Do chỉ dùng 1 chip PIC 8259A nên các đường tín hiệu Cas0, Cas1, Cas2 không sử dụng. PIC không dùng đệm kênh nên tín hiệu SP/EN được cố định lên mức 1 (nối với Vcc). Các thiết bị ngoại vi từ NV0 đến NV7 được nối với các đầu vào từ IR0 đến IR7. Riêng phần giải mã chọn chip được thực hiện trên IC 74138. Kênh địa chỉ của 74138 có 3 bit ABC được nối tương ứng với các bit A1A2A3 của kênh địa chỉ của hệ vi xử lý. Chân E1 của 74138 được cố định lên mức 1 (nối với Vcc). Chân /E3 của 74138 được nối tín hiệu AEN còn chân /E2 là đầu ra của tổ hợp A15-A4 theo mạch OR. Với tổ chức

của 74138 như vậy, tín hiệu chip select /CS cho 8259 là 0000h cho các thanh ghi bên trong PIC có địa chỉ chẵn và 0001h cho các thanh ghi bên trong PIC có địa chỉ lẻ.

Kênh dữ liệu 8 bit của PIC 8259A được nối với byte thấp của kênh dữ liệu hệ vi xử lý 80286

Hình 6.17. **Ghép PIC 8259A với hệ vi xử lý 80X86**

6.4.2. Lập trình điều khiển hoạt động cho chip 8259A

Sử dụng sơ đồ ghép nối trên hình 6.17 để làm cơ sở cho lập trình điều khiển hoạt động cho chip 8259A.

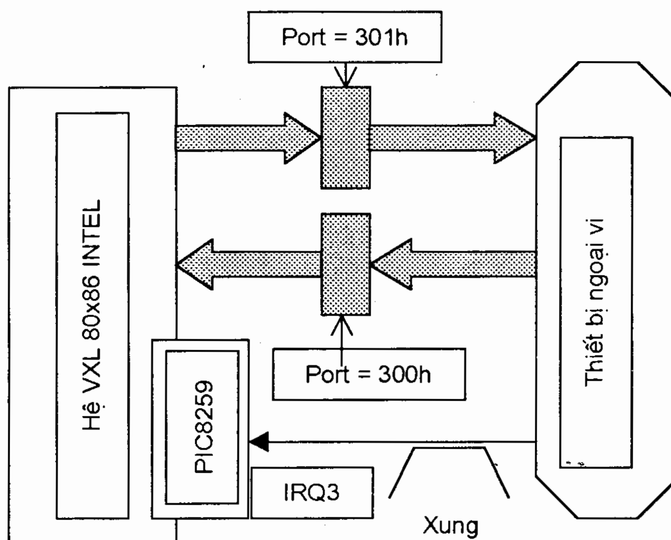
Giả thiết rằng hệ vi xử lý làm việc với các thiết bị ngoại vi NV0 đến NV7 được nối trực tiếp với các đầu vào IR0 đến IR7 của PIC 8259 và PIC làm việc ở chế độ ưu tiên cố định (là chế độ thông thường trong tổ chức ngắt của các hệ vi xử lý chuyên dụng) với lệnh kết thúc ngắt EOI là loại thường.

Các thiết bị ngoại vi NV0 đến NV7 kích hoạt chip PIC bằng mức xung chứ không phải bằng sườn xung.

Trong bảng vectơ ngắt của hệ vi xử lý số hiệu cơ sở của PIC này là 70h (có nghĩa là IR0 có số hiệu ngắt = 70h, IR1 có số hiệu ngắt = 71h,..., IR7 có số hiệu ngắt = 77h).

Bây giờ phải lập trình khởi tạo cho PIC và đưa vào hoạt động với giả thiết rằng hệ này đang làm các công việc nội tại như tính toán, xử lý, gia công hay biến đổi dạng tín hiệu bên trong, nhưng khi có yêu cầu ngắt ở chân IR3 (ngoại vi NV3 có yêu cầu được phục vụ) thì thực hiện chức năng đọc 1Kbyte dữ liệu dạng mã BCD từ cổng 300h rồi đổi thành 2 byte mã ASCII (mỗi byte chứa 1 chữ số thập phân) rồi đưa ra cổng 301H. Khi hoàn tất chức năng này phải quay về trao quyền cho chương trình chính.

Giải: Sơ đồ chức năng của hệ vi xử lý này có thể mô tả như sau



Chương trình khởi tạo và điều khiển chức năng hoạt động của hệ theo yêu cầu được xây dựng như sau:

Modul 1: Cài đặt vector ngắt 73h cho chương trình con phục vụ ngắt IR3_SERVICE vào bảng vector ngắt:

```
MOV AX,0000H   ; Chuyển địa chỉ mảng chứa bảng vectơ ngắt
MOV ES,AX      ;vào thanh ghi ES
MOV DX,OFFSET IR3_SERVICE ;lấy địa chỉ offset của
MOV WORD PTR ES:[73H*4],DX ; chương trình con PV ngắt
                               ;chuyển vào vectơ ngắt 73h
MOV DX,CS ;lấy địa chỉ Seg của CTCPV ngắt IR3_SERVICE
MOV WORD PTR ES:[73H*4+2],DX ;chuyển vào vectơ ngắt 73h
```

Modul 2: Khởi tạo PIC 8259A theo cấu hình yêu cầu:

```
CLI  ; cấm mọi ngắt (trừ ngắt NMI)
```

```
MOV AL,00011111b  ;icw1= mức,4 byte,1 chip, CPU 16/32 bit
OUT PIC0,AL  ; ghi vào ICW1 của chip PIC
MOV AL,70H  ; icw2= số hiệu ngắt đầu tiên =70h
OUT PIC1,AL  ; ghi vào ICW2 của chip PIC
MOV AL,00000001b  ;icw4= ưu tiên thường, không đệm, EOI
                  ;thường, 80x86
OUT PIC1,AL        ; ghi vào ICW4 của chip PIC
MOV AL,0000000b   ;ocw1= mặt nạ=0 - cho tất cả các IRi được
                  ;hoạt động.
OUT PIC0, AL      ; ghi vào OCW2 của chip PIC
STI ; cho phép các ngắt hoạt động trở lại
```

Modul 3: Viết chương trình con phục vụ ngắt IR3_SERVICE:

```
  PUSH AX
  PUSH BX
  PUSH CX
  PUSH DX
  IN AL, PIC1; đọc vào nội dung thanh ghi mặt nạ ngắt
  MOV AH,AL;tạm giữ trong AH
  MOV AL,0F7H; mã lệnh chỉ cho phép IR3 tác động
  OUT PIC1,AL;đưa ra thanh ghi lệnh OCW1của PIC
  MOV CX,1024; CX là bộ đếm 1kb
LOOP_1Kb:
  MOV DX, PortIn
  IN AL,DX  ; nhận mã BCD từ cổng PortIn
  MOV BL,AL;gửi tạm vào BL
  PUSH CX; cất CX
  MOV CL,4; xử lý chữ số BCD cao trước
  SHR AL,CL; dịch phải 4 bit : d7d6d5d4→d3d2d1d0
  ADD AL,30H; cộng mã ASCII
  MOV DX,PortOut
  OUT DX,AL; xuất ra cổng ra PortOut chữ số này
  MOV AL,BL; lấy lại giá trị cũ
  AND AL,0FH;lọc bỏ 4 bit cao để xử lý chữ số BCD thấp
```

```
ADD AL,30H; cộng với mã ASCII
OUT DX,AL; xuất ra cổng ra PortOut chữ số này
POP CX; khôi phục lại CX
LOOP LOOP_1Kb;lặp lại 1024 lần
MOV AL, AH;lấy lại nội dung cũ của mặt nạ ngắt
OUT PIC1, AL;trả lại vào OCW1 của PIC
MOV AL, 20H; chuẩn bị mã lệnh kết thúc ngắt EOI(=20h)
OUT PIC0,AL;ghi và OCW2 của PIC
POP DX
POP CX
POP BX
POP AX
IRET
```

Ghép các modul chương trình lại với nhau ta được chương trình đầy đủ như sau:

```
;-----------------------------------------------------------------------
;Khai báo các hằng:
;Địa chỉ mạch ngắt ưu tiên 8259 PIC0= 0000h
;Địa chỉ mạch ngắt ưu tiên 8259 PIC1= 0001h
;-----------------------------------------------------------------------
        PIC0 EQU 0000H;  A0=0(icw1, ocw2, ocw3)
        PIC1 EQU 0001H;  A0=1(icw2, icw3, icw4, ocw1)
        PortIn EQU 300H; Địa chỉ cổng vào
        PortOut EQU 301H; Địa chỉ cổng ra
;-----------------------------------------------------------------------
CODE_SEG SEGMENT            ; Mở mảng mã lệnh
ASSUME CS:CODE_SEG, DS:CODE_SEG
    ORG 100H   ;Tạo file dạng COM cho chương trình MONITOR
;-----------------------------------------------------------------------
; Trong thủ tục chính phải cài được vector ngắt 73H
; vào bảng vector ngắt hệ vi xử lý
;-----------------------------------------------------------------------
MAIN PROC  ;Thủ tục chính
```

```
MOV AX,0000H  ; Chuyển địa chỉ mảng chứa bảng vectơ ngắt
MOV ES,AX        ;vào thanh ghi ES
MOV DX,OFFSET IR3_SERVICE ;lấy địa chỉ offset của
MOV WORD PTR ES:[73H*4],DX ; chương trình con PV ngắt
                                    ;chuyển vào vectơ ngắt 70h
MOV DX,CS            ;lấy địa chỉ Seg của
                     ;chương trình con PV ngắtIR3
MOV WORD PTR ES:[73H*4+2],DX ;chuyển vào vectơ ngắt 73h
CALL INIT  ;gọi thủ tục khởi tạo chip PIC 8259
;##########################################################
; Các lệnh của chương trình chính main Proc đặt ở đây
;##########################################################
MAIN ENDP
;-------------------------------------------------------------------------------
IR3_SERVICE PROC  ; chương trình con PV ngắt IR3
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        IN AL, PIC1; đọc vào nội dung thanh ghi mặt nạ ngắt
        MOV AH,AL;tạm giữ trong AH
        MOV AL,0F7H; mã lệnh chỉ cho phép IR3 tác động
        OUT PIC1,AL;đưa ra thanh ghi lệnh OCW1của PIC
        MOV CX,1024; CX là bộ đếm 1Kb
LOOP_1Kb:
        MOV DX, PortIn
        IN AL,DX  ; nhận mã BCD từ cổng PortIn
        MOV BL,AL;gửi tạm vào BL
        PUSH CX; cất CX
        MOV CL,4; xử lý chữ số BCD cao trước
        SHR AL,CL; dịch phải 4 bit : d7d6d5d4→d3d2d1d0
        ADD AL,30H; cộng mã ASCII
        MOV DX,PortOut
```

```
        OUT DX,AL; xuất ra cổng ra PortOut chữ số này
        MOV AL,BL; lấy lại giá trị cũ
        AND AL,0FH;lọc bỏ 4 bit cao để xử lý chữ số BCD thấp
        ADD AL,30H; cộng với mã ASCII
        OUT DX,AL; xuất ra cổng ra PortOut chữ số này
        POP CX; khôi phục lại CX
        LOOP LOOP_1Kb;lặp lại 1024 lần
        MOV AL, AH;lấy lại nội dung cũ của mặt nạ ngắt
        OUT PIC1, AL;trả lại vào OCW1 của PIC
        MOV AL, 20H; chuẩn bị mã lệnh kết thúc ngắt EOI(=20h)
        OUT PIC0,AL;ghi và OCW2 của PIC
        POP DX
        POP CX
        POP BX
        POP AX
IRET
IR3_SERVICE ENDP
;-------------------------------------------
INIT PROC ; chương trình khởi tạo cho chip PIC 8259
        PUSH AX
        PUSH DX
        CLI  ; cấm mọi ngắt (trừ ngắt NMI)
        MOV AL,00011111b   ;icw1= mức,4 byte,1 chip, CPU 16/32 bit
        OUT PIC0,AL   ; ghi vào ICW1 của chip PIC

        MOV AL,70H         ; icw2= số hiệu ngắt đầu tiên =70h
        OUT PIC1,AL         ; ghi vào ICW2 của chip PIC

        MOV AL,00000001b   ;icw4= ưu tiên thường, không đệm, EOI
                           ;thường, 80x86
        OUT PIC1,AL            ; ghi vào ICW4 của chip PIC

        MOV AL,0000000b   ;ocw1= mặt nạ=0 - cho tất cả các IRi được
                          ;hoạt động.
```

```
        OUT PIC0, AL          ; ghi vào OCW2 của chip PIC
        STI; cho phép các ngắt hoạt động trở lại
        POP DX
        POP AX
        RET
    INIT ENDP
    ;-------------------------------------------
    CODE_SEG ENDS       ;đóng mảng mã lệnh
    END MAIN            ;kết thúc chương trình
```

Trong chương trình, sau khi cài đặt chương trình con phục vụ ngắt IR3 vào bảng vectơ ngắt tại địa chỉ SEG:OFF = 0000:73h*4 thì thủ tục chính gọi chương trình con INIT nhằm khởi đầu cho PIC 8259A.Vì chương trình chính khi hoạt động có thể thay đổi mặt nạ ngắt của nên tại chương trình con trong khi phục vụ yêu cầu ngắt IR3 phải lưu trữ giá trị của mặt nạ ngắt hiện hành rồi dùng lệnh để che các yêu cầu ngắt khác, chỉ để yêu cầu ngắt IR_3 được nhận biết. Sau khi xử lý xong chức năng của mình, chương trình con IR3_SERVICE phải trả lại giá trị mặt nạ ngắt cũ cho 8259A rồi mới trở về trao quyền điều khiển cho chương trình chính MAIN PROC.

6.4.3. Nối tầng chip 8259A

Nối tầng PIC 8259A với hệ vi xử lý 80286 được trình bầy trên hình 6.18. Các tín hiệu ghép nối phần lớn là tương thích nhau. Do dùng 2 chip PIC 8259A để nối tầng nên các đường tín hiệu Cas0, Cas1, Cas2 được sử dụng. PIC chủ không dùng đệm kênh nên tín hiệu SP/EN được cố định lên mức 1 (nối với Vcc). Các thiết bị ngoại vi từ NV0 đến NV7 được nối với các đầu vào từ IR0 đến IR7PIC thợ không dùng đệm kênh nhưng tín hiệu SP/EN được dùng để qui chiếu PIC nên nó được cố định ở mức 0 (nối GND). Các thiết bị ngoại vi từ NV8 đến NV15 được nối với các đầu vào từ IR0 đến IR7 của PIC thợ. Đầu ra INTR của PIC thợ được nối với đầu vào IRQ2 của PIC chủ. Riêng phần giải mã chọn chip được thực hiện trên IC 74138. Kênh địa chỉ của 74138 có 3 bit ABC được nối tương ứng với các bit A1A2A3 của kênh địa chỉ của hệ vi xử lý. Chân E1 của 74138 được cố định lên mức 1 (nối với Vcc). Chân /E3 của 74138 được nối tín hiệu AEN còn chân /E2 là đầu ra của tổ hợp A15-A4 theo mạch OR. Với tổ chức của 74138 như vậy, tín hiệu chip select /CS cho 8259 chủ là 0000h cho các thanh ghi bên trong PIC có địa chỉ chẵn và 0001h cho các thanh ghi bên trong PIC có địa chỉ lẻ.

Hình 6.18. **Nối tầng PIC 8259A với hệ vi xử lý 80x86**

Tín hiệu chip select /CS cho 8259 thợ lúc này là 000Eh cho các thanh ghi bên trong PIC có địa chỉ chẵn và 000Fh cho các thanh ghi bên tron PIC có địa chỉ lẻ. Cho rằng vector ngắt của PIC chủ đặt trong bảng vector ngắt từ ngắt số hiệu 40h đến 47h còn vector ngắt của PIC thợ đặt trong bảng vector ngắt từ ngắt số hiệu 70h đến 77h.Theo sơ đồ trên thì các lệnh khởi đầu cho các thanh ghi bên trong của PIC chủ và PIC thợ sẽ là:

Cho PIC chủ (nạp lệnh qua cổng 0000h và 0001h):

ICW1 = 0001 0001h; sườn xung, cho PIC chủ, ICW4.

ICW2 = 40h ; số hiệu ngắt 40h cho IR0, IR1 = 41h, IR7 = 47h.

ICW3 = 04h ; PIC chủ nối với PIC thợ ở IRQ2.

ICW4 = 01h ; PIC chủ không dùng đệm, CPU = 80 × 86.

Cho PIC thợ (nạp lệnh qua cổng 000Eh và 000Fh):

ICW1 = 0001 0001h; sườn xung, cho PIC thợ, ICW4.

ICW2 = 70h ; số hiệu ngắt 70h cho IR0, IR1 = 71h, IR7 = 77h.

ICW3 = 02h ; PIC thợ nối với PIC chủ ở IRQ2.

ICW4 = 01h ; PIC chủ không dùng đệm, CPU = 80 × 86.

Chương 7
TRUYỀN THÔNG TIN NỐI TIẾP

7.1. CÁC KHÁI NIỆM VỀ TRUYỀN SỐ LIỆU

7.1.1. Mạng thông tin truyền số liệu

Trong kỹ thuật truyền số liệu thì thông tin số trong hệ vi xử lý phải qua một loạt biến đổi để đảm bảo thông tin đó truyền đi xa (hay nhận từ xa về) mà không bị biến dạng, tức là bảo đảm tính trung thực của bản tin gốc. Để có thể thực hiện được các biến đổi cần thiết cho thông tin trong mạng cần nắm được những khái niệm cơ bản của kỹ thuật truyền số liệu.

Tin tức

Là tập hợp các ký hiệu mang nội dung thông tin như bức điện báo, tệp dữ liệu, bức ảnh, công văn...Để truyền các kí hiệu cụ thể đó phải chuyển chúng thành tín hiệu điện.

Tín hiệu số liệu

Tín hiệu số liệu là đại lượng vật lý mang nội dung tin tức (dạng điện áp u, dòng điện i ...). Tín hiệu có thể chia làm 2 thể loại là tín hiệu tương tự (Analog) và tín hiệu số (Digital).

Tín hiệu tương tự là dạng tín hiệu phổ biến nhất trong tự nhiên mà sự thay đổi tham số đặc trưng của nó (u hoặc i hoặc tham số đặc trưng khác) là liên tục theo thời gian.

Tín hiệu số là hệ quả của tín hiệu tương tự sau khi qua các công đoạn: rời rạc hoá theo thời gian, lượng tử theo mức, mã hoá bằng bộ mã nhị phân nào đó. Bit là đơn vi thông tin cơ sở của thông tin số. Mỗi bit chỉ có thể nhận giá trị hoặc là 0 hoặc là 1.

Thông tin truyền số liệu ngày nay chủ yếu sử dụng tín hiệu số làm cơ sở để hình thành mọi dạng tin tức.

Các tham số của hệ thống truyền số liệu

- Tốc độ số liệu

  Là số lượng bit thông tin truyền trong một đơn vị thời gian.

  $$V_x = \frac{1}{t_x}[bit/s], \quad t_x\text{: độ rộng 1 bit tin.}$$

- Tốc độ đường truyền

  Là số lượng đơn vị tín hiệu truyền trong một đơn vị thời gian.

  $$N = \frac{1}{t_o}[Baud], \quad t_0\text{: độ rộng 1 đơn vị tín hiệu.}$$

  Tuỳ theo cách mã hoá thông tin mà một Baud có thể có một hay nhiều bit thông tin.

- Khả năng thông qua của kênh

  Là tốc độ số liệu lớn nhất có thể đạt được với kênh truyền số liệu đó.

  $$C = V_{x\,(max)} \quad [bit/s]$$

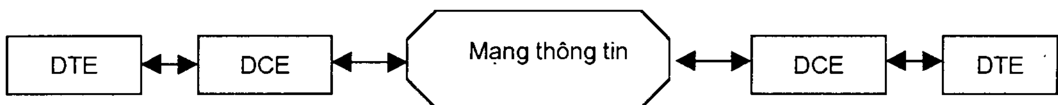
- Lượng tin tức

  Một phần tử tin có *m* mức thì có *m* thông tin. Do vậy *n* phần tử *m* mức thì có m^n lượng thông tin.

  Ví dụ: Tín hiệu nhị phân có m = 2 nên nếu có n phần tử nhị phân ta sẽ có: $I = 2^n$ lượng thông tin.

Tổ chức của hệ thống thông tin truyền số liệu

Hệ thống truyền tin số có thể được mô tả theo sơ đồ khối như hình 7.1.



***Hình 7.1.* Sơ đồ khối hệ thống truyền tin số**

Các thành phần chính trong hệ gồm:

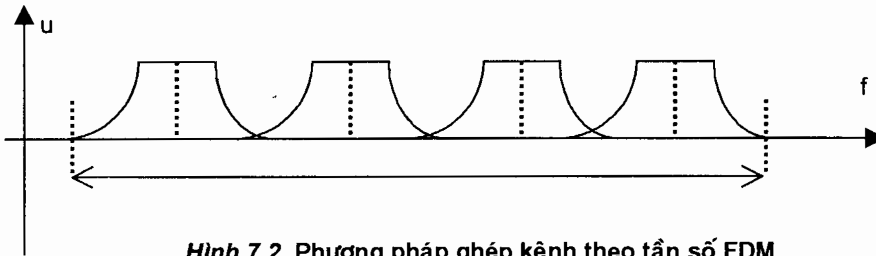
♦ DTE (*Data Terminal Equipment - Thiết bị đầu cuối truyền số liệu*) là tập hợp nguồn thông tin số, các mạch logic xử lý, gia công, biến đổi tin. Các ký hiệu tin tức trước khi truyền được DTE mã hoá thành các tổ hợp mã rồi chuyển thành các tín hiệu điện tương ứng (các xung điện) để truyền tới DCE. Trong DTE thực hiện hai quá trình mã hoá, quá trình thứ nhất là biến ký hiệu tin tức thành tổ hợp mã đơn giản trong đó các phần tử mã đều là các phần tử mang tin, quá trình này bắt buộc phải xảy ra. Quá trình thứ hai là thêm vào các từ mã đơn giản một số các phần tử kiểm tra để phát hiện hoặc sửa lỗi của thông tin. Quá trình

này có thể thực hiện hay không tuỳ theo yêu cầu về chất lượng thông tin của từng hệ thống. DTE có thể là hệ vi xử lý chuyên dụng hay máy tính hay mạng máy tính.

- DCE (*Data Communication Equipment - Thiết bị truyền dẫn số liệu*). DCE có thể là Modem hay các thiết bị biến đổi tin khác có chức năng biến đổi dãy tín hiệu số thành dãy tín hiệu có điều chế cho phù hợp với kênh truyền. Tín hiệu sau khi đã điều chế, được đưa tới hệ thống thông tin, ở đây có các thiết bị ghép kênh để truyền tín hiệu theo một kênh đó.

Có nhiều nguyên lý ghép kênh khác nhau như ghép kênh theo tần số, ghép kênh theo thời gian.

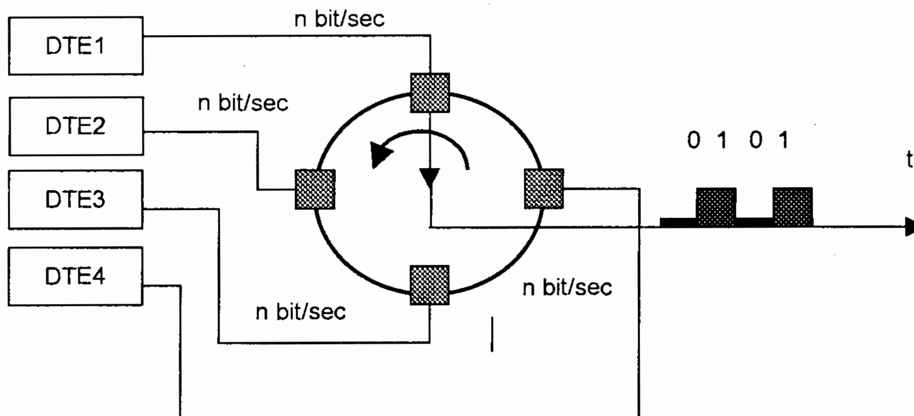
Ghép kênh theo tần số (FDM: Frequency Division Multiplexing). Theo phương pháp này (hình 7.2) giữa các kênh phải có khe hở để tránh xuyên âm. Phương pháp FDM cho tốc độ bit cỡ 2000 bit/s, phù hợp cho truyền dẫn tốc độ thấp.



Hình 7.2. Phương pháp ghép kênh theo tần số FDM

Ghép kênh theo thời gian (TDM: Time Division Multiplexing). Trong hệ TDM, thời gian được chia thành các khung, một khe thời gian cho một kênh. Hệ thống PCM của mạng thoại có 32 khe thời gian (32 kênh). Ghép kênh TDM có thể thực hiện tại lớp bit hoặc lớp ký tự.

Ví dụ, 4 DTE ghép kênh theo bit được biểu diễn trên hình 7.3. Chu kỳ ghép gồm 4 khe thời gian bằng nhau được chia đều cho 4 đầu cuối (mỗi khe cho một kênh). Kết quả ta có một dòng bit chung ở đầu ra nối tiếp.



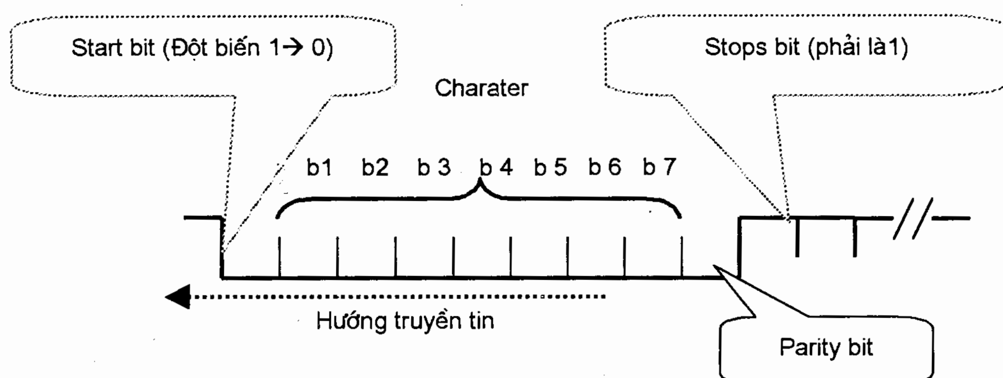
Hình 7.3. Phương pháp ghép kênh theo thời gian TDM

7.1.2. Các phương pháp truyền tin số

Có 2 phương pháp truyền dẫn tín hiệu số chủ yếu đó là phương pháp truyền không đồng bộ và phương pháp truyền đồng bộ.

Phương pháp truyền tín hiệu không đồng bộ

Trong phương pháp truyền tín hiệu không đồng bộ *(Asynchronous Communication)* thì các nhóm bit (tương ứng các ký hiệu) được truyền đi theo một khung tin độc lập, mỗi khung tin được bắt đầu và kết thúc bằng các bit đặc biệt (bit Start, bit Stop) với mục đích tạo đồng bộ giữa bên thu và bên phát. Thời điểm bắt đầu truyền các nhóm bit là bất kỳ và độc lập nhau. Bit Start luôn luôn phải là 0. Bit Stop có thể là 1 bit, 1+1/2 bit hoặc 2 bit với giá trị luôn luôn phải là 1. Bit Parity có thể được cài vào để thực hiện phát hiện sai sơ bộ cho bên thu. Tuỳ theo loại mã sử dụng mà số lượng bit trong khung tin có thể là 5, là 6, là 7 hay là 8 (hình 7.4).



Hình 7.4. **Phương pháp truyền tin không đồng bộ**

Phương pháp này có ưu điểm là yêu cầu đồng bộ giữa thu và phát không đòi hỏi chặt chẽ nhờ có bit Start và bit Stop xác định thời điểm đầu và cuối của nhóm bit cho nên sự sai pha tích luỹ chỉ diễn ra trong thời gian thu nhóm bit đó. Chính điều đó dẫn tới ưu điểm là thiết bị trong hệ thống khá đơn giản, giá thành hệ thống thấp.

Nhược điểm của phương pháp truyền không đồng bộ là hiệu quả sử dụng kênh thấp do phải truyền nhiều bit Start và bit Stop là những bít không mang tin. Mặt khác, tốc độ truyền tin cũng bị hạn chế. Các Modem có tốc độ không lớn thường sử dụng phương pháp này.

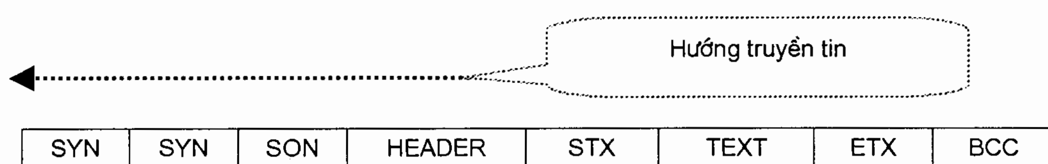
Tốc độ truyền dữ liệu theo phương pháp nối tiếp được đo bằng baud/s. Với dữ liệu chỉ có 2 mức (0 và 1) và mỗi thay đổi mức tín hiệu chỉ mã hoá 1 bit thì có thể hiểu baud/s = bit/s. Các giá trị tốc độ truyền thường gặp trong thực tế là 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200 baud...

Phương pháp truyền tín hiệu đồng bộ

Phương pháp truyền tín hiệu đồng bộ (Synchronous Communication) khắc phục được các nhược điểm của phương pháp không đồng bộ. Bản chất của phương pháp này là các tín hiệu số được gửi đi một cách liên tục với tốc độ không đổi. Trong trường hợp này, thiết bị thu đầu cuối cần phải tạo ra và duy trì tần số nhịp đồng bộ với tín hiệu số đầu vào (tức là đồng bộ với tần số nhịp bên phát) trong suốt thời gian làm việc.

Có nhiều phương pháp để duy trì đồng bộ giữa thu và phát, như chèn thêm các bít đồng bộ vào dãy tín hiệu số, thiết lập mã truyền dẫn đặc biệt ...

Các thuật toán trên cho phép duy trì đồng bộ giữa thu và phát ngay cả khi dãy tín hiệu gồm một chuỗi bit 0 hay bit 1 hoặc thiết bị phát tạm dừng.



Hình 7.5. **Khuôn dạng khối tin của giao thức BISYNC trong truyền tin đồng bộ.**

Trong hệ thống truyền đồng bộ, số liệu có thể được tổ chức thành từng khối (Block), theo các giao thức khác nhau (Protocol). Ví dụ giao thức chuẩn BISYNC (Binary Synchronous Communication Protocol) có khuôn dạng của một khối tin BISYNC như mô tả trên hình 7.5. Trong đó

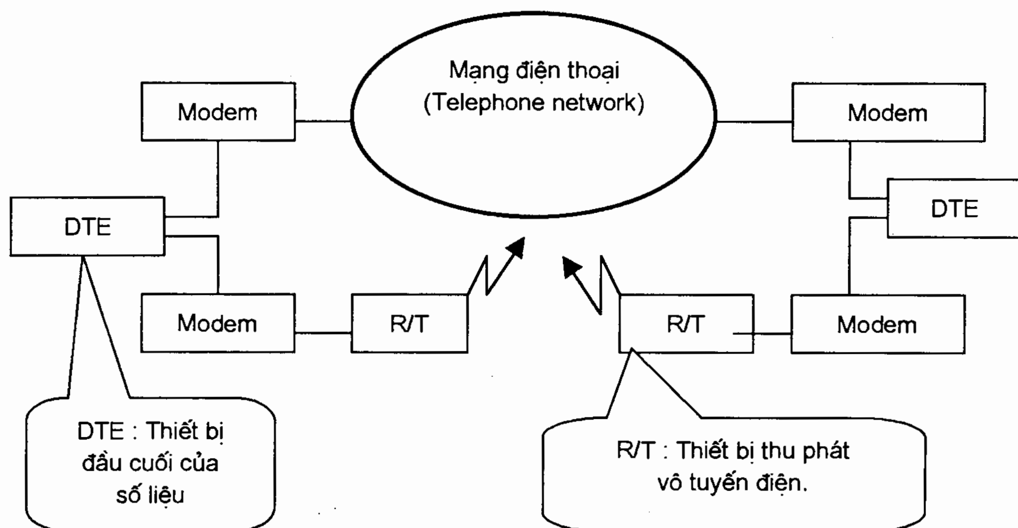
- SYN : Ký tự đồng bộ. Sau khi phát hiện hai ký tự đặc biệt dùng làm tín hiệu đồng bộ đã biết trước, thiết bị thu bắt đầu ghi nhận ký tự SOH.
- SOH (Start of Header) là byte mở đầu, xác định kích thước và các đặc tính của trường Header.
- HEADER: Trường này có độ dài thay đổi, có thể dùng để chứa địa chỉ nơi nhận tin.
- STX (Start of Text) chỉ ra rằng ngay sau byte này là phần bắt đầu của văn bản.
- TEXT: Trường này có độ dài thay đổi, chứa đựng các ký tự mã ASCII hoặc EBCDIC (là nội dung tin tức cần truyền).
- ETX (End of Text) là byte đánh dấu sự kết thúc của khối tin văn bản.
- BCC (Block Check Character) là khối tổng kiểm tra (CRC) dùng để kiểm tra phát hiện lỗi.

Truyền số liệu qua mạng điện thoại và hệ thống thông tin vô tuyến

Mạng điện thoại công cộng (Puplic Telephone Network) là hệ thống thông tin được ứng dụng rộng rãi nhất với quy mô rộng lớn ở mọi quốc gia. Chức năng chính của nó là đảm bảo kết nối và truyền dẫn các cuộc thoại ở các phạm vi khác nhau.

Việc ứng dụng rộng rãi các hệ thống vi xử lý trong mọi lĩnh vực đời sống, kinh tế đặt ra một vấn đề bức thiết là tổ chức hệ thống thông tin số với các quy mô khác nhau. Đối với các hệ thống thu phát thông tin hiện nay, phương án thông dụng nhất là sử dụng chính mạng điện thoại công cộng và các máy thông tin vô tuyến (ở đây là các máy thông tin sóng ngắn và cực ngắn) với các thiết bị ghép nối thích hợp đó là các Modem .

Sơ đồ tổng quát nối ghép thiết bị đầu cuối số liệu với mạng điện thoại và mạng vô tuyến (hình 7.6).



***Hình 7.6.* Mạng điện thoại và mạng vô tuyến trong thông tin truyền số liệu**

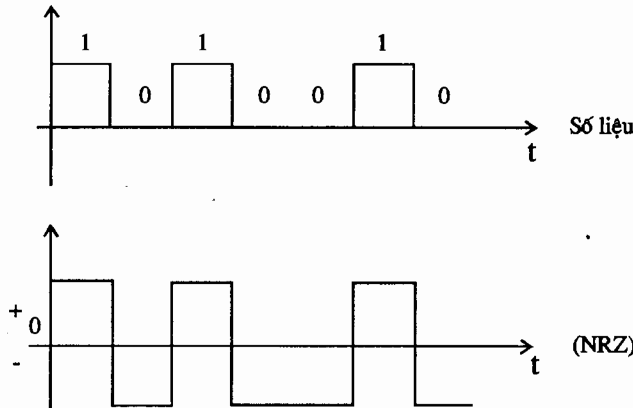
MODEM với chức năng phối hợp biến đổi dạng tín hiệu số thành dạng tín hiệu tương tự phù hợp với kênh thoại khi truyền tin và biến đổi ngược lại khi nhận tin, hai quá trình đó là điều chế và giải điều chế (Modulation - Demodulation gọi tắt là MODEM). MODEM đảm bảo việc truyền dẫn số liệu nên người ta thường gọi là thiết bị truyền dẫn số liệu (Data Communication Equipment - DCE). Tuỳ theo thuật toán làm việc và mức độ thông minh của MODEM mà tốc độ truyền tin cũng như chất lượng thông tin có thể được cải thiện rất nhiều.

Các MODEM có các phương thức cho truyền song công, đơn công, không đồng bộ (tốc độ thấp), đồng bộ (tốc độ cao). Cùng một MODEM có thể chuyển đổi được các tốc độ khác nhau.

7.1.3. Một số dạng mã thông dụng trong truyền số liệu

Dạng mã cơ sở là dạng mã hoá lại tín hiệu ban đầu sao cho thành phần một chiều bị loại bỏ (tổng đại số các xung dương và âm bằng không). Sau đây là một số loại mã phổ biến.

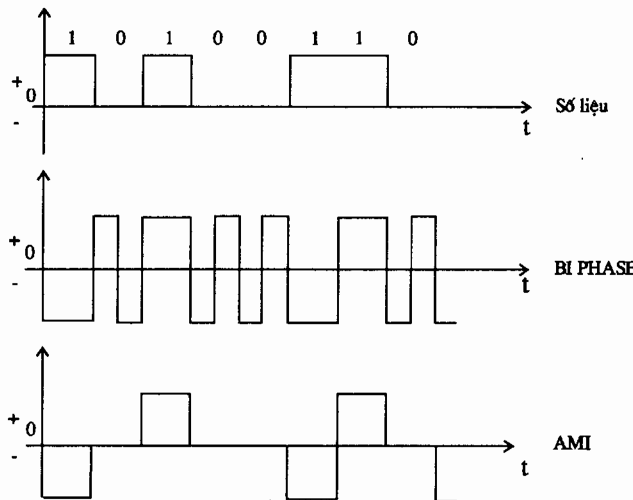
***a. Mã NRZ** (Non Return to Zero)* được biểu diễn trên hình 7.7. Mã *NRZ* nghĩa là "không trở về không". Các giá trị logic "1" và "0" ở trong tín hiệu ban đầu được mã hoá thành tín hiệu "dương" và "âm". Mã NRZ chỉ dùng trong thiết bị số chứ không làm mã đường truyền.



Hình 7.7. Mã NRZ (Non Return to Zero)

b. Mã BIPHASE - mã 2 pha được biểu diễn trên hình 7.8. Mã này có sự thay đổi cực tính giữa các bit đối với logic "0" (sự đổi cực phụ được chèn thêm ở giữa bit).

c. Mã AMI (Altenet Mark Inversion) là mã đảo dấu biến đổi, nó được biểu diễn trên hình 7.8. Mã này làm việc với 3 mức: "0", "-", "+" . Các bit liên tục được đảo dấu xen kẽ nhau. Số xung "+" bằng số xung "-", để triệt tiêu thành phần một chiều.



***Hình 7.8.* Mã BiPhase và AMI**

Mã đường dây

Để đảm bảo truyền tin giữa bên phát và bên thu với chất lượng cao, vấn đề mã hoá và phát hiện lỗi trên đương truyền tín hiệu là vấn đề quan trọng, dưới đây là một số loại mã đường dây.

a. Mã ADI (hình 7.9)

Mỗi bit thay đổi đều được đảo, tránh được một chuỗi dài các số "0" liên tục trong tín hiệu gốc, bằng cách thêm "về 0" cho ADI, không một trạng thái "mở" nào kéo dài hơn một nửa độ rộng xung (tại 50% chu kỳ) một loạt các số "1" trở thành các xung được đánh dấu rõ ràng.

b. Mã AMI. (đã trình bày ở trên)

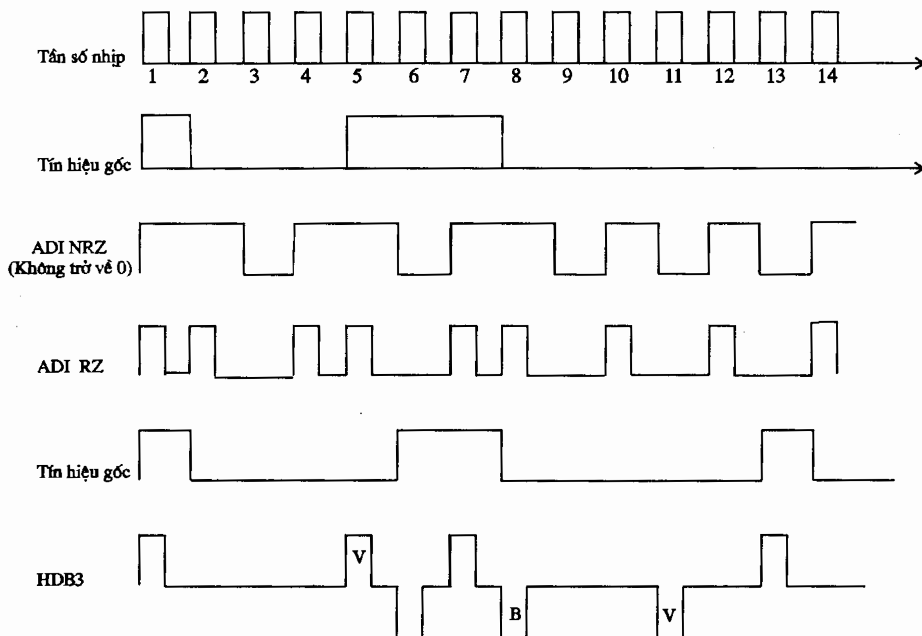
c. Mã HDB3 - High Density Bipolar code (hình 7.9)

Trên cơ sở mã HDBn, với n = 3 chính là mã dựa trên cơ sở mã AMI với quy tắc: Các bit "0" kế tiếp nhau không quá 3.

Ví dụ: Một nhóm 4 bit "0" liên tiếp sẽ thay bằng một nhóm 3 bit "0", bít "0" cuối cùng thay bằng bit "V" (0000 → 000V, xung "V" có cực tính giống cực tính xung ngay trước nhóm đó. Giữa hai xung V liên tiếp nhau phải có số lẻ các xung, nếu chưa đảm bảo thì phải thêm vào 1 xung "B" sao cho "B" tham gia vào quy tắc đảo xung xen kẽ.

Đặc điểm mã này không tồn tại thành phần một chiều, dễ tách xung đồng bộ, khả năng phát hiện lỗi tốt.

d. Mã BIPHASE (đã trình bày ở trên)



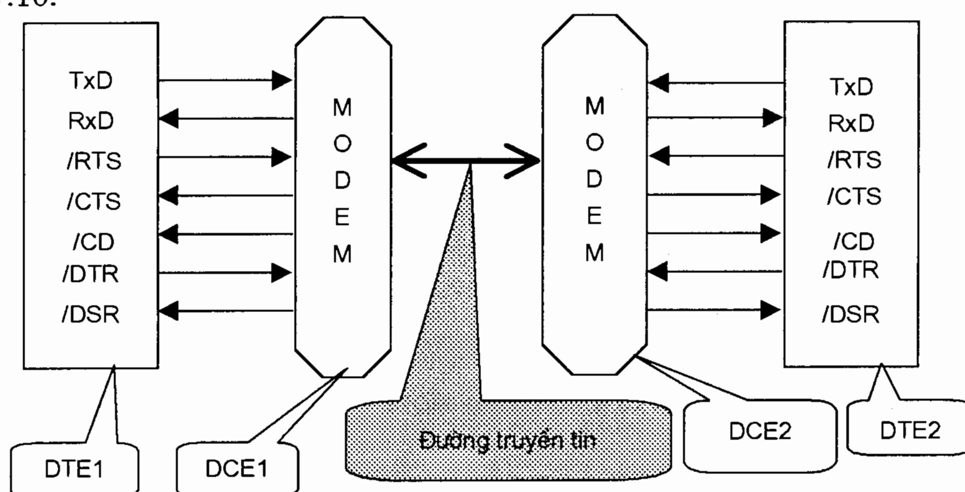
Hình 7.9. **Các loại mã đường dây**

7.2. TỔ CHỨC ĐƯỜNG TRUYỀN TÍN HIỆU NỐI TIẾP

Để tổ chức đường truyền tin nối tiếp với hệ vi xử lý và để giảm tối đa các mạch phụ thêm ở bên ngoài, công nghệ vi điện tử đã chế tạo ra các vi mạch tổ hợp cỡ lớn LSI lập trình được, có khả năng thực hiện phần lớn các chức năng truyền dẫn và phối ghép với hệ vi xử lý. Đó là mạch thu phát dị bộ vạn năng IN8250/16450 của hãng National Semiconductor UART(universal asynchronous receiver trasnmitter chip) và mạch thu phát đồng bộ - dị bộ vạn năng 8251A USART (universal synchronous and asynchronous receiver trasnmitter chip).

Với các mạch phối ghép như trên, việc truyền tin dị bộ chẳng hạn sẽ được thực hiện nhờ một UART ở đầu phát và một UART khác ở đầu thu. Khi có ký tự để phát, 8250 (8251A) tạo ra khung cho ký tự bằng cách gắn thêm vào mã ký tự các bit start, parity và stop rồi gửi liên tiếp từng bit ra đường truyền. Bên phía thu, một 8250 (8251A) khác sẽ nhận ký tự, tháo bỏ khung, kiểm tra party, rồi chuyển sang dạng song song để CPU đọc.

Các tín hiệu nối tiếp với dạng xung điện áp thể hiện giá trị logic 0 và 1 được đưa ra hệ thống như trên hình 7.6 có thể truyền được thông tin đi xa trên đường điện thoại với dải tần hạn chế. Ở đầu phát, các xung 0, 1 phải được điều chế thành các xung âm tần để truyền đi. Còn ở đầu thu các xung âm tần phải được giải điều chế để khôi phục lại tín hiệu số ban đầu. Thiết bị hoàn thành phần việc chính là MODEM. Vì vậy, để có thể truyền thông tin nối tiếp giữa các thiết bị ta cần nối ghép chúng với nhau theo cách thức như được biểu diễn trên hình 7.10.

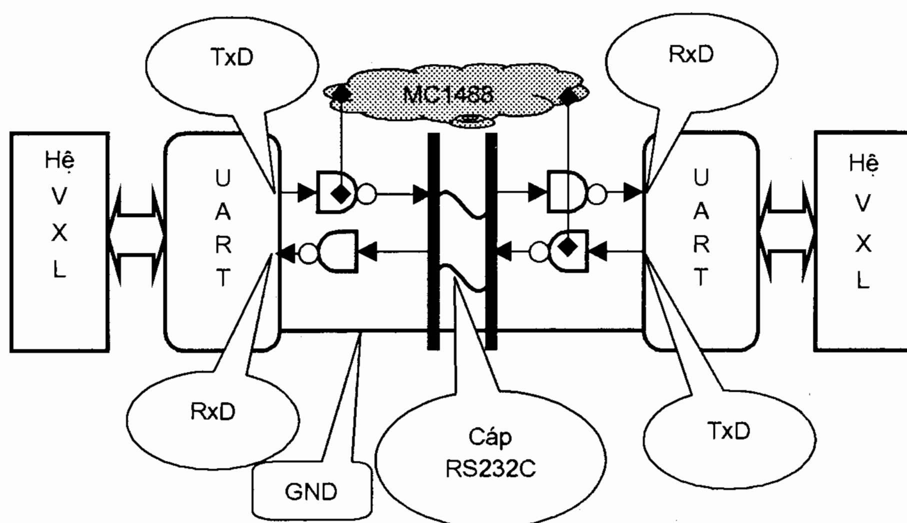


Hình 7.10. **Ghép UART với MODEM**

Các tín hiệu trên hình 7.10 là một phần của các tín hiệu dùng trong chuẩn phối ghép quốc tế trong truyền dữ liệu RS 232C. Hoạt động của mạch truyền dữ liệu theo sơ đồ này như sau:

- Lúc bắt đầu làm việc DTE1 đưa ra xung /DTR (Data Terminal Ready)=0 để báo cho DCE1 biết là nó sẵn sàng làm việc. DCE1 cũng vậy, khi bắt đầu làm việc nó đưa ra xung /DSR (Data Set Ready) = 0 để báo cho DTE1 biết là nó sẵn sàng thu phát thông tin. Sau đó DCE1 được điều khiển để phát xung gọi sang phía bên kia để kết nối liên lạc.
- Nếu DTE2 sẵn sàng làm việc nó sẽ gửi trả lời sang phía bên DTE1 những xung âm tần theo quy ước.
- Khi DTE1 có ký tự cần gửi, nó đưa ra đầu /RTS (Request To Send) = 0. DCE1 sẽ đưa ra /CD (Carrier Detect) = 0 để báo là đường truyền đã thông và khi nó sẵn sàng nhận dữ liệu để chuyển đi thì nó sẽ đưa ra /CTS (Clear To Send) = 0. Nhận được thông báo này DTE1 gửi các ký tự sang DCE2 qua đầu phát TxD. Sau khi truyền xong ký tự này, DTE1 đưa ra /RTS = 1 để báo kết thúc. DCE1 cũng kết thúc công việc và đưa ra /CTS = 1 để thông báo.

Trong thực tế giữa TDE và DCE thường có các dây cáp và đầu nối cáp theo tiêu chuẩn RS 232C. Trên hình 7.11 là ví dụ sơ đồ nối ghép giữa hệ vi xử lý-UART với MODEM qua các đầu cáp chuẩn RS 232C bằng các bộ khuyếch đại đường dây phát MC 1488 và khuếch đại đường dây thu MC 1489.



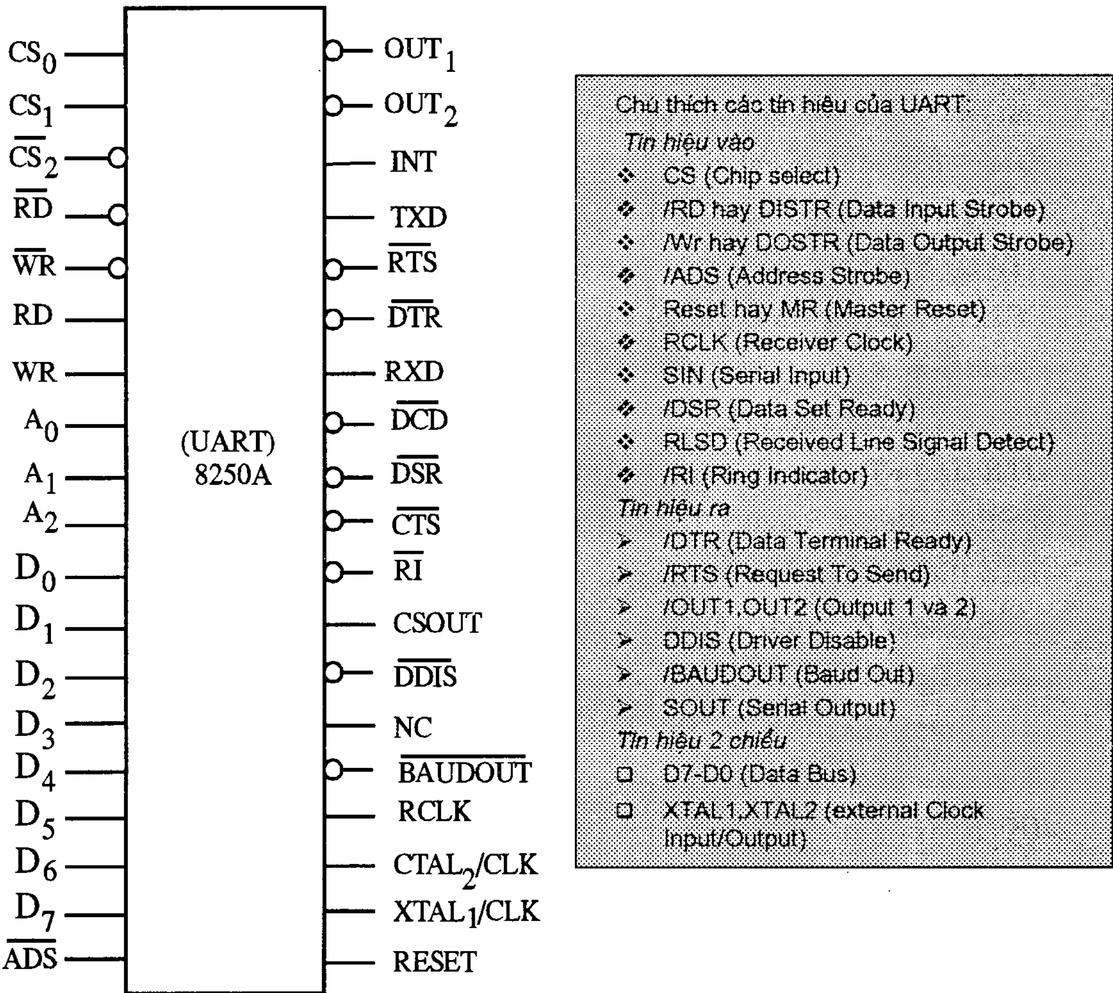
Hình 7.11. **Nối ghép giữa hệ vi xử lý - UART với MODEM qua các đầu cáp chuẩn RS 232C.**

7.3. MẠCH THU PHÁT DỊ BỘ VẠN NĂNG IN8250A/16450

UART 8250A (Universal Asynchronous Receiver/Transmitter) là mạch phối ghép vào ra không đồng bộ, lập trình được dùng để chuyển thông tin song song (từ máy tính) thành nối tiếp (ra đường truyền) và ngược lại. Nó là chip IC chuyên dụng loại LSI 40 chân. Số lượng thanh ghi bên trong của UART 8250A nhiều hơn của USART 8251A, nên UART 8250A được dùng phổ biến hơn.

7.3.1. Tổ chức của UART 8250A

Sơ đồ chân tín hiệu chip UART 8250A được trình bầy trên hình 7.12. Các tín hiệu được trình bầy theo nhóm.



Hình 7.12. **Sơ đồ chân tín hiệu chip UART 8250A**

- CS_0, CS_1, $/CS_2$ là các chân tín hiệu chip select, nếu chúng ở mức tích cực thì UART được hệ vi xử lý chọn (kích hoạt).

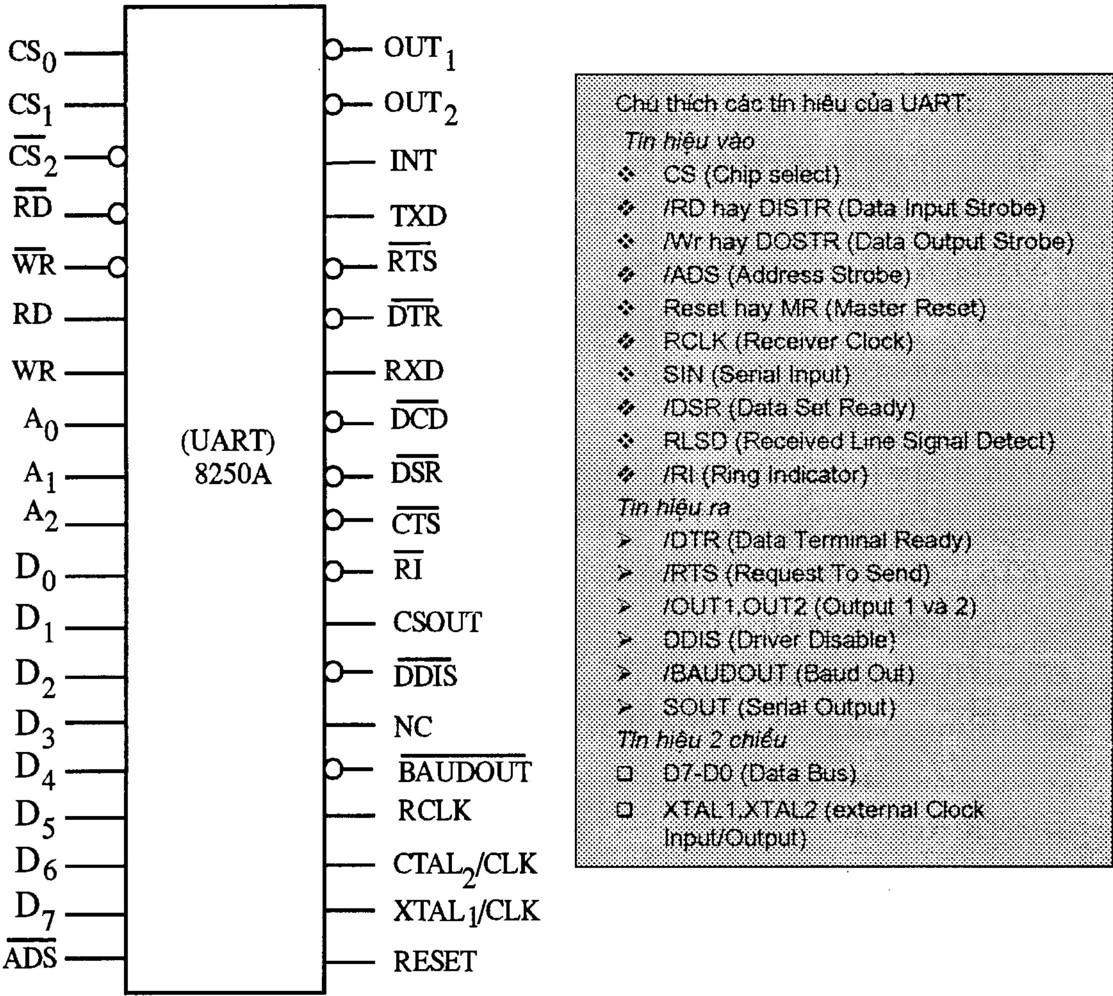
- A_0, A_1, A_2 là các chân tín hiệu địa chỉ cho phép chọn 10 thanh ghi bên trong của 8250A.

- CSOUT (Chip Select Out) ở mức cao cho biết hệ vi xử lý đã chọn vi mạch 8250A bằng CS_0, CS_1/CS_2.

- /ADS (Address Select): ở mức thấp thông báo rằng rằng tín hiệu địa chỉ và các tín hiệu chọn chíp đã ổn định.

7.3.1. Tổ chức của UART 8250A

Sơ đồ chân tín hiệu chip UART 8250A được trình bầy trên hình 7.12. Các tín hiệu được trình bầy theo nhóm.



Hình 7.12. **Sơ đồ chân tín hiệu chip UART 8250A**

- CS_0, CS_1, $/CS_2$ là các chân tín hiệu chip select, nếu chúng ở mức tích cực thì UART được hệ vi xử lý chọn (kích hoạt).

- A_0, A_1, A_2 là các chân tín hiệu địa chỉ cho phép chọn 10 thanh ghi bên trong của 8250A.

- CSOUT (Chip Select Out) ở mức cao cho biết hệ vi xử lý đã chọn vi mạch 8250A bằng CS_0, CS_1/CS_2.

- /ADS (Address Select): ở mức thấp thông báo rằng rằng tín hiệu địa chỉ và các tín hiệu chọn chíp đã ổn định.

Nhóm tín hiệu số liệu:

- $D_0 \div D_7$: Kênh dữ liệu vào/ra 2 chiều.
- SIN (Serial In): Lối vào cổng nối tiếp của số liệu từ ngoại vi.
- /WR (/DOSTR) hay /DOUTS: các lối vào để hệ vi xử lý ghi số liệu hoặc lệnh điều khiển vào các thanh ghi bên trong của 8250A.
- /RD (RISTR) hay /DINS: các lối vào để hệ vi xử lý đọc số liệu hoặc từ trạng thái của các thanh ghi bên trong của 8250A.
- /DDIS ở mức thấp chỉ vi xử lý đang đọc số liệu từ 8250A.

Nhóm tín hiệu nhịp

- $XTAL_1$, $XTAL_2$ là các lối vào của phần tử dao động thạch anh.
- RCLK (Receiver Clock) là lối vào xung đồng hồ.
- /BaudOut là lối ra xung nhịp.

Nhóm điều khiển MODEM

- DTR (Data Terminal Ready) : Thiết bị đầu cuối sẵn sàng.
- /DSR (Data Set Ready): Thiết bị truyền thông sẵn sàng.
- /RTS (Request to Send): yêu cầu phát.
- /CTS (Clear to Send): sẵn sàng nhận.
- /DCD (Detect Carrier Data): Phát hiện sóng mang.
- /RI (Ring Indicator): Chỉ thị chuông.

Nhóm điều khiển khác

- /INTR: Yêu cầu ngắt.
- /OUT2, /OUT1: Dùng cho người sử dụng cho các ứng dụng không phải chuẩn RST232.
- V_{CC}, GND, Reset: Nguồn nuôi, đất chung và tín hiệu khởi động.

Sơ đồ khối chức năng của chip UART 8250A được trình bày trên hình 7.13.

Khối điều khiển ngắt:

Logic điều khiển ngắt.

Thanh ghi cho phép ngắt.

Thanh ghi nhận dạng ngắt.

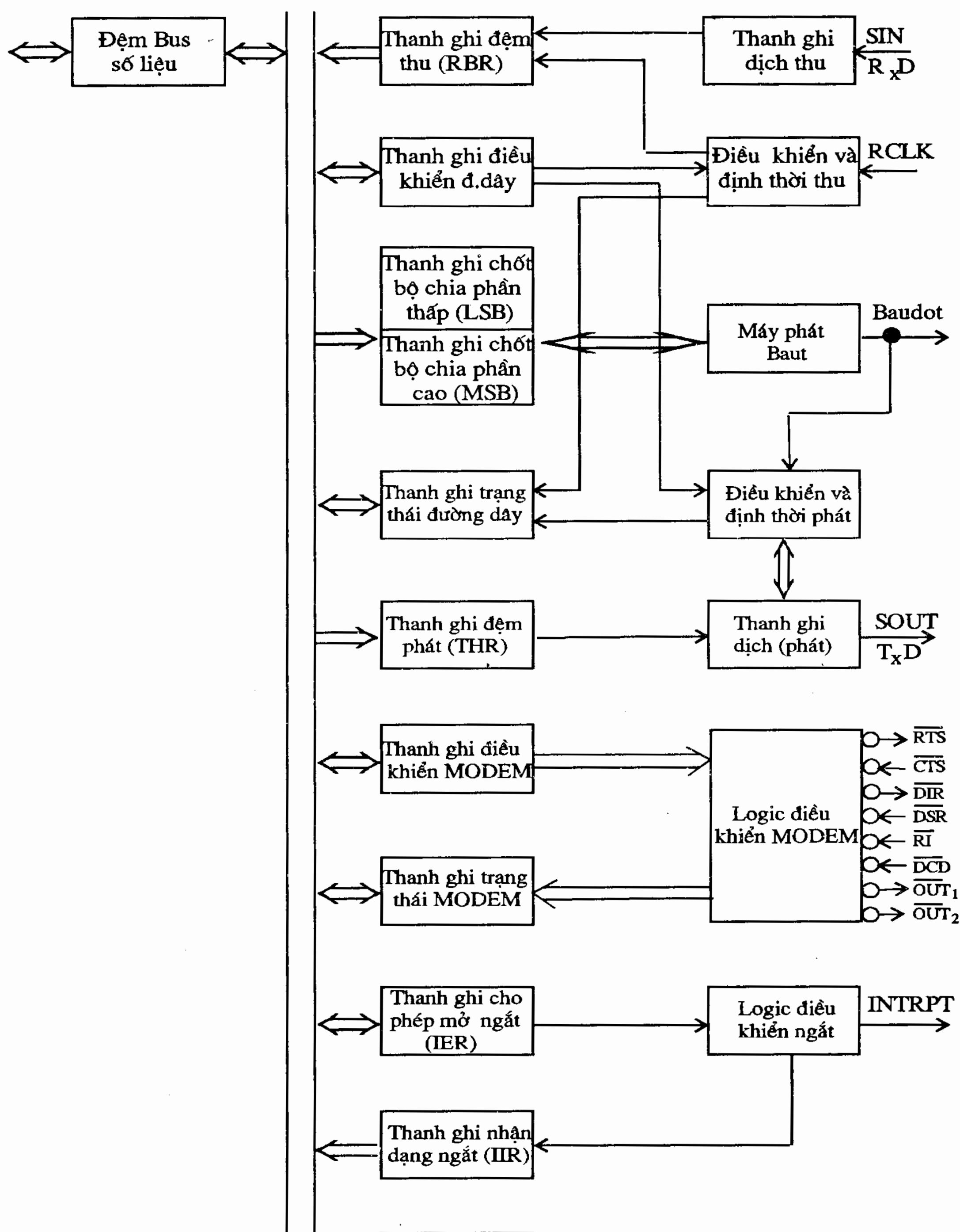
Khối phát tín hiệu:

Bộ tạo BAUD.

Các bộ chia tần số LSB, MSB để tạo tốc độ phát.

Điều khiển và định thời gian cho khối phát.

Điều khiển và định thời gian cho khối thu.



Hình 7.13. **Sơ đồ khối của chip UART 8250A**

Khối điều khiển MODEM.

Logic điều khiển MODEM.

Thanh ghi điều khiển MODEM.

Khối thu:

Thanh ghi dịch thu.

Thanh ghi đệm thu.

Khối phát:

Thanh ghi dịch phát.

Thanh ghi đệm thu.

Khối điều khiển chung:

Thanh ghi điều khiển đường dây.

Thanh ghi trạng thái đường dây.

7.3.2. Các thanh ghi bên trong của 8250 *(bảng 7.1)*

Bảng 7.1

DLAB	A_2	A_1	A_0	Thanh ghi được chọn
0	0	0	0	Thanh ghi đệm thu (RBR)+thanh ghi đệm phát (THR).
0	0	0	1	Thanh ghi cho phép ngắt (IER).
1	0	0	0	Thanh ghi chốt/ bộ chia phần thấp (LSB).
1	0	0	1	Thanh ghi chốt/ bộ chia phần cao (MSB).
x	0	1	0	Thanh ghi nhận dạng ngắt (IIR).
x	0	1	1	Thanh ghi điều khiển đường dây (LCR).
x	1	0	0	Thanh ghi điều khiển MODEM (MCR).
x	1	0	1	Thanh ghi trạng thái đường dây (LSR).
x	1	1	0	Thanh ghi trạng thái MODEM (MSR).
x	1	1	1	Thanh ghi nháp (SPR).

UART 8250A có 3 tín hiệu A_2, A_1, A_0 cùng tín hiệu DLAD (Divisor Latch Access Bit - Bit truy cập bộ chia tần số) để chọn ra các thanh ghi bên của 8250A.

✧ *Thanh ghi điều khiển đường truyền (Line Control Reggister - LCR).* Dạng thức của thanh ghi LCR.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DLAB	SBCB	SP	EPS	PEN	STB	WLS_1	WLS_0

Thanh ghi trạng thái đường truyền cho biết trạng thái của việc truyền tín hiệu trên đường dây. Các tín hiệu THRE, BI, FE, PE, OE, RxDR đều là các nguyên nhân gây ngắt nếu các bit cho phép tương ứng trong thanh ghi IER được

lập. Nó là thanh ghi thông báo trạng thái đường truyền - phát hiện Break BI(dòng tin bị ngắt), các lỗi máy thu (tràn khung FE, chẵn lẻ PE, tràn số OE, số liệu sẵn sàng và trạng thái không có số liệu truyền).

Dạng thức LSR như sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	TSRE	THRE	BI	FE	PE	OE	RxDR

+ D_0 (R_xDR - Receiver Data Ready - số liệu thu sẵn sàng).

1: Đã nhận được một ký tự và để nó trong thanh ghi đệm thu (RBR).

0: Khi CPU đọc thanh ghi RBR.

+ D_1 (OE - Overrun error - lỗi do thu đè).

1: Có hiện tượng thu đè (có thể do CPU bị chậm).

0: Khi CPU đọc thanh ghi LSR.

+ D_2: (PE - Parity error - lỗi parity).

1: Có lỗi Parity.

0: Khi CPU đọc LSR.

+ D_3: (FE - Framing error - lỗi khung).

1: Có lỗi khung (ví dụ: Bit Stop = 0).

0: Khi CPU đọc LSR.

+ D_4: (BI - Break Interrupt có sự gián đoạn trong khi truyền).

1: Khi tín hiệu ở đầu vào phần thu ở mức thấp lâu hơn thời gian giành cho một kí tự.

0: Khi CPU đọc LSR.

+ D_5: (THRE - Transmiter Holding Register Empty).

1: Ký tự đã được chuyển từ THR sang TSR.

0: Khi CPU đưa kí tự tới thanh ghi THR.

+ D_6: (TSRE - Transmitter Shift Register Empty - thanh ghi dịch phát rỗng).

1: Khi một ký tự đã được phát đi.

0: Khi có một ký tự được chuyển từ THR sang TSR.

+ D_7: Luôn luôn bằng 0.

✧ *Thanh ghi đệm phát (Transmitter Holding Register - THR).*

Ký tự cần phát đi phải được lấy từ CPU ghi vào thanh ghi này với bit DLAB = 0, sau đó UART 8250A tạo khung tin cho nó như đã định và đưa từng bit ra chân tín hiệu Sout.

✧ *Thanh ghi đệm thu (Receiver Buffer Register - RBR).*

Khi 8250A nhận được một ký tự qua chân tín hiệu SIN, nó tháo bỏ khung cho ký tự và lưu ký tự tại thanh ghi đệm thu để CPU đọc. CPU đọc được ký tự trong thanh ghi này khi bit DLAB = 0.

✧ *Thanh ghi cho phép ngắt (Interrupt Enable Register - IER).*

Thanh ghi này dùng để cho phép hay cấm các nguồn gây ngắt khác nhau. Trong khi UART 8250A hoạt động, có thể tác động tới CPU thông qua chân INTRPT của UART để tạo ra 4 loại ngắt. Mỗi bit trong số D_3, D_2, D_1, D_0 ở mức cao sẽ cho phép các hiện tượng tương ứng với bit đó được đưa ra yêu cầu ngắt đối với CPU.

Dạng thức IER như sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	MODEM	RLINE	T_xEMTY	R_xRDY

Bốn bit cao (D_7, D_6, D_5, D_4) luôn bằng 0.

+ D_0 = 1: Cho phép gây ngắt khi đệm thu đầy (R_xD đầy).

+ D_1 = 1: Cho phép gây ngắt khi đệm phát rỗng (T_xD rỗng).

+ D_2 = 1: Cho phép các tín hiệu trạng thái đường dây thu gây ngắt.

+ D_3 = 1: Cho phép ngắt từ MODEM.

✧ *Thanh ghi nhận dạng ngắt (Interrupt Identification Register - IIR).*

Thanh ghi này được đọc mỗi khi có yêu cầu ngắt từ UART 8250A. CPU đọc bit D_0 của thanh ghi này để biết có yêu cầu ngắt và kiểm tra bit D_2 D_1 D_0 để xác định nguồn gốc của yêu cầu ngắt.

Khi UART bị Reset, chỉ có ngắt ưu tiên 1 được phục vụ, tuy nhiên có thể thay đổi điều này bằng cách dùng mặt nạ che đi các yêu cầu ngắt nào đó.

Dạng thức IIR như sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	ID_2	ID_1	ID_0

Mức ưu tiên cho các ngắt được thể hiện ở bảng 7.2.

Bảng 7.2

ID_2	ID_1	ID_0	Mức ưu tiên	Tên loại ngắt	Nguồn gốc	ID_i bị xoá khi
0	0	1	0			
1	1	0	1	Trạng thái đường thu	Lỗi khung, thu đè, lỗi Parity, gián đoạn khi thu	Đọc LSR
1	0	0	2	Đệm thu đầy	Đệm thu đầy	Đọc RBR
0	1	0	3	Đệm phát rỗng	Đệm phát rỗng	Đọc IIR, ghi THR
0	0	0	4	Trạng thái MODEM	CTS, DSR, RI, RLSD	Đọc MSR

Các bít $D_3 \div D_7$ luôn bằng 0.

✧ *Thanh ghi điều khiển MODEM (MODEM Control Register - MCR)*

Thanh ghi này điều khiển các tín hiệu ra của MODEM, nó cho phép điều khiển các tín hiệu tại các chân /DTR và /RTS của UART.

Dạng thức MCR như sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	LOOP	OUT_2	OUT_1	RTS	DTR

D_0 = DTR = 1 tích cực hoá chân ra /DTR của 8250A (=0).

= 0: thụ động hoá chân ra /DTR của 8250A (=1).

D_1 = RTS = 1: tích cực hoá chân ra /RTS của 8250A (=0).

= 0: thụ động hoá chân ra /RTS của 8250A (=1).

D_2 D_3 : Điều khiển các đầu ra phụ OUT1 OUT2. Cụ thể

D1 = 1 → OUT1 = 0 và ngược lại.

D2 = 1 → OUT2 = 0 và ngược lại.

D_4 = 1: Cho phép điều khiển 8250A làm việc ở chế độ nối vòng cục bộ để kiểm tra chức năng của UART.

- SOUT = 1.
- SIN: Không nối với bên ngoài.
- Các thanh ghi dịch của phần phát và phần thu nối vòng với nhau.
- Các chân điều khiển vào của MODEM (/DSR, /CTS, RI, RLSD) không được nối ra ngoài mà được nối liền trong mạch với các chân điều khiển ra của MODEM (/DTR, RTS, OUT1, OUT2).

✧ *Thanh ghi trạng thái MODEM (MODEM Status Register - MSR).*

Thanh ghi này còn được gọi là thanh ghi trạng thái vào của RS 232C, nó cho biết trạng thái hiện thời của các tín hiệu điều khiển MODEM từ đường dây.

Dạng thức của MSR như sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
RLSD	RI	DSR	CTS	ΔRLSD	ΔRI	ΔDSR	ΔCTS

Các bit của MSR có dấu (Δ) để nói rằng UART 8250A đang hoạt động, nên có sự thay đổi của các tín hiệu đó thì các bit tương ứng sẽ được lập, riêng tín hiệu RI thì ΔRI ám chỉ sự thay đổi từ mức thấp lên mức cao.

Các bít $D_0 \div D_3$ phản ánh sự thay đổi trạng thái các chân RS 232C, giá trị 1 của các bít này cho thấy cửa vào của nó đã bị thay đổi từ lần đọc sau cùng. Quá trình đọc thanh ghi này sẽ xoá tất cả các bít $D_0 \div D_3$.

+ D_0 = 1 - DCTS (Delta Clear to Send) đã thay đổi trạng thái.

+ D_1 = 1 - DDSR (Delta Data Set Ready) đã thay đổi trạng thái.

+ D_2 = 1 - DDCD (Delta Data Carrier Detect) đã thay đổi.

Các bit $D_4 \div D_7$ phản ánh các tín hiệu vào từ cổng RS 232C.

+ D_4 = 1 - CTS (Clear to Send) là tích cực.

+ D_5 = 1 - DSR (Data Set Ready) là tích cực.

+ D_6 = 1 - RI (Ring Indicator) là tích cực.

+ D_7 = 1 - DCD (Data Carrier Detect) là tích cực.

✧ *Bộ chia tần tạo tốc độ truyền tin.*

Bộ chia này gồm 2byte là LSB và MSB, dùng để chia tần số xung nhip để tạo tốc độ *baud* mong muốn. Hệ số chia được tính theo công thức:

$$\text{Hệ số chia} = \frac{\text{Tần số nhịp chuẩn}}{\text{Tốc độ Baud yêu cầu} \times 16}$$

Ví dụ, tần số nhịp do bộ tạo xung thạch anh của 8250A phát ra là 1,8432 MHz. Tốc độ Baud theo yêu cầu là tốc độ của luồng tin phát và thu ở các chân SOUT và SIN. Để có tốc độ truyền là 50 Baud thì hệ số chia (Hexa Decimal) phải là 900.

Việc ghi hệ số chia 900 vào LSB và MSB được tiến hành như sau:

Bước 1. Gán bit DALB (D_7 = 1) vào thanh ghi số liệu có địa chỉ offset 3 để ghi vào các bộ chia.

Bước 2. Ghi giá trị 00h vào chốt bộ chia LSB - địa chỉ offset 0.

Bước 3. Ghi giá trị 09h vào chốt bộ chia MSB - địa chỉ offset 1.

XTAL là các chân để cắm thạch anh dùng cho mạch dao động bên trong của UART 8250A. Trường hợp sử dụng tín hiệu đồng hồ chuẩn từ bên ngoài thì đưa xung đồng hồ vào chân $XTAL_1$ thì bỏ lửng chân $XTAL_2$. Chân RLSD (Receiver Line Signal Detect) là chân báo phát hiện thấy sóng mang, tức đã thiết lập được đường truyền. Tín hiệu /RI là tín hiệu chuẩn của RS 232C, để MODEM thông báo là có chuông reo. Tín hiệu INTRPT là tín hiệu để yêu cầu ngắt CPU trong trường hợp 8250A được lập trình để có khả năng gây ngắt CPU.

Khởi động chế độ phát cho UART 8250A

Khởi động chế độ phát cho 8250A theo trình tự:

+ Ghi vào thanh ghi số liệu (địa chỉ offset = 3) để:

D_7 = DLAB = 1 để chuẩn bị thao tác với bộ chia tốc tần.

Khung tin với số bit Stop theo D_2 (1; 1,5 hoặc 2).

Số bit số liệu (5, 6,7, 8) theo D_2, D_1.

+ Ghi giá trị bộ chia tần số vào các thanh ghi của bộ chia LSB (offset = 0), MSB (offset = 1), tuỳ theo giá trị tốc độ.

+ Ghi giá trị cho phép ngắt vào thanh ghi ngắt.

Phát số liệu nối tiếp

Phát số liệu nối tiếp theo trình tự:

– Đọc thanh ghi nhận dạng ngắt để biết bộ đệm có rỗng không để có thể phát tin.

– Ghi vào thanh ghi điều khiển MODEM để đưa DTR (bit D_0 = 1), RTS (D_1 = 1) điều khiển MODEM chuẩn bị phát.

– Đọc thanh ghi trạng thái MODEM để kiểm tra các bit DSR (D_5), RI (D_6), DCD (D_7) xem đã chuẩn bị sẵn sàng chưa.

– Đọc thanh ghi trạng thái đường dây để xem có sai số không và 2 thanh ghi truyền và đệm có rỗng không để đưa tin ra.

– Nạp vào thanh ghi đệm phát ký tự cần phát.

Thu số liệu nối tiếp

Thu số liệu nối tiếp theo trình tự:

– Gán DTR (D_0 = 1) vào thanh ghi điều khiển MODEM.

– Đọc trạng thái MODEM DSR (D_1 = 1), RI (D_6 = 1), DCS (D_7 = 1) ở thanh ghi trạng thái MODEM.

– Đọc thanh ghi trạng thái đường dây để biết đã có số liệu thu chưa (D_0 = 1 - R_xRDY) hoặc đọc thanh ghi nhận diện ngắt để biết thêm số liệu thu được.

– Đọc số liệu vào từ thanh ghi đệm số liệu vào hệ vi xử lý.

Kiểm tra sự hoạt động của UART 8250A

Kiểm tra sự hoạt động của 8250A theo trình tự sau:

– Ghi vào thanh ghi điều khiển MODEM các bít $D_0 = D_1 = D_2 = D_3 = D_4 = 1$ để điều khiển MODEM và cách nối vòng như đã trình bày ở thanh ghi điều khiển lối ra RS 232C.

– Ghi số liệu vào thanh ghi đệm phát.

– Đọc số liệu từ thanh ghi đệm thu.

– So sánh 2 số liệu ghi ra và đọc vào, nếu bằng nhau thì UART 8250A hoạt động đúng.

7.3.3. Nối ghép UART 8250A với hệ vi xử lý

Hình 7.14. **Nối ghép UART 8250A với hệ vi xử lý**

Các đường tín hiệu liên kết giữa 8250 với vi xử lý và MODEM được trình bầy như trên hình 7.14.

Kênh dữ liệu: $D_0 \div D_7$ từ 8250A được nối đến hệ vi xử lý qua cổng đệm như qua mạch 74245 chẳng hạn. Hướng truyền tin của nó được điều khiển bằng tổ hợp /IORD, /IOWR của hệ vi xử lý và /CS của UART 8250A.

Tín hiệu ngắt INTR của các 8250A sẽ nối đến đầu ngắt của bộ vi xử lý hay đầu vào IRi của của PIC 8259 qua tín hiệu khống chế OUT2.

Nhóm tín hiệu nối với MODEM để điều khiển và khống chế thu phát tin dưới dạng nối tiếp. Có thể kiểm tra UART bằng cách nối vòng đầu phát TxD với đầu thu RxD qua tổ hợp *Opto Coupler* và *nút ấn*.

Nhóm tín hiệu địa chỉ A_0, A_1, A_2 và chọn chip CS được nối với các tín hiệu tương ứng của hệ vi xử lý để khống chế đọc, ghi 8250A và chọn các thanh ghi bên trong 8250A theo địa chỉ chẳng hạn từ 3F8h đến.

Tín hiệu nhịp dùng cho 8250A lấy chung từ bộ dao động ngoài nên chân XTAL1 không cần sử dụng. Tốc độ thu phát do bộ chia tần quyết định và do vậy đầu ra BaudOut được nối với RCLK.

7.3.4. Lập trình cho UART 8250A

Bài tập 1. Khởi tạo cổng truyền thông có địa chỉ cổng nối tiếp UART 8250 là ComPort = 378h, địa chỉ thanh ghi bộ chia tần thấp LSB = 378h, địa chỉ thanh ghi bộ chia tần cao MSB = 379h, địa chỉ thanh ghi LCR = 37Bh. Khung dữ liệu thu/phát = 8 bit DATA, 2 bit STOP, không dùng PARITY.

Giải

Cho rằng các địa chỉ của cổng và các thanh ghi được khai báo ở đầu chương trình:

```
        ComPort  EQU 378h;
        LSB EQU 378h  ;
        MSB EQU 379h  ;
        LCR EQU 37Bh ;
```

thì thủ tục khởi tạo UART có dạng:

```
        INIT PROC
        PUSH AX
        PUSH DX
        PUSH DI
;------------- khởi tạo cổng truyền thông ---------------------;
            MOV      DX, LCR        ; DX chứa OFFSET của thanh ghi LCR
            MOV   AL, 80H    ; Bit DLAB = 1 cho phép truy nhập bộ
            OUT   DX, AL     ; chia để tính tốc độ Baud
;------------ xác định tốc độ truyền dữ liệu -------------------;
```

```
        MOV   DI, OFFSET Table; DI chứa OFFSET bảng hệ số chia
        ADD   DI, 0; trỏ tới 110 baud rate
        MOV   DX, LSB; DX chứa OFFSET của thanh
        MOV   AL, CS:[DI]      ; lấy phần thấp của hệ số chia
        OUT   DX, AL           ; LSB chứa phần thấp của hệ số chia
        MOV   DX, MSB
        MOV   AL, CS:[DI]+1    ; lấy phần cao của hệ số chia
        OUT   DX, AL           ; MSB chứa phần cao của hệ số chia
        MOV   DX,  LCR    ; DX chứa OFFSET của thanh ghi LCR
        MOV   AL, 00000111b  ; các tham số khởi tạo cho AL
        OUT   DX, AL        ;  LCR chứa tham số khởi tạo về số bit
                             ; trong 1 từ, parity, số bít stop
        POP DI
        POP DX
        POP AX
        RET
Table:  DW 1047;  110 tốc độ baud rate
        DW 768;  150 tốc độ baud rate
        DW 348;  300 tốc độ baud rate
        DW 192;  600 tốc độ baud rate
        DW 96;  1200 tốc độ baud rate
        DW 48;  2400 tốc độ baud rate
        DW 24;  4800 tốc độ baud rate
        DW 12;  9600 tốc độ baud rate
INIT ENDP
;-----------------------------------------------------
```

Bài tập 2. Viết chương trình con phục vụ ngắt để phát 1 byte Data_Trans qua cổng truyền thông nối tiếp UART 8250 có địa chỉ cổng là ComPort = 378h, địa chỉ thanh ghi MCR = 37Ch, địa chỉ thanh ghi LSR = 37Dh, địa chỉ thanh ghi MSR = 37Eh.

Giải.

Cho rằng các địa chỉ của cổng và các thanh ghi được khai báo ở đầu chương trình:

```
  ComPort  EQU 378h;
  MCR EQU 37Ch;
  LSR EQU 37Dh;
  MSR EQU 37Eh;
```

thì thủ tục điều khiển phát 1 byte có dạng:

```
TransCOM   PROC   ; chương trình con phát 1 byte
  PUSH AX
  PUSH DX
  MOV DX, MCR ;
  MOV AL, 03H  ; gán DTR(bit0)=RTS(bit1)=1→thanh ghi đ.kh MODEM
  OUT DX, AL
  MOV DX, MSR
  IN AL,DX
  AND AL, 30h
  CMP AL, 30H;     trong MSR: DSR(bit5)=CTS(bit4)=1?
  JNZ  Exit_
  MOV DX, LSR
  IN AL, DX
  AND AL, 60h
  CMP AL, 60H          ; trong LSR: TSRE(bit6)=THRE(bit5)=1?
  JNZ  Exit_
  MOV DX, ComPort ; phát byte qua cổng COM
  MOV AL, Data_Trans
  OUT DX, AL
Exit_:  POP DX
        POP AX
        IRET
TransCOM   ENDP
;---------------------------------------------------
```

Bài tập 3. Viết chương trình con phục vụ ngắt để thu 1 byte dữ liệu từ cổng truyền thông nối tiếp UART 8250 có địa chỉ cổng là ComPort = 378h, địa chỉ thanh ghi MCR = 37Ch, địa chỉ thanh ghi LSR = 37Dh, địa chỉ thanh ghi MSR = 37Eh. Kết quả đặt vào biến Data_Receive.

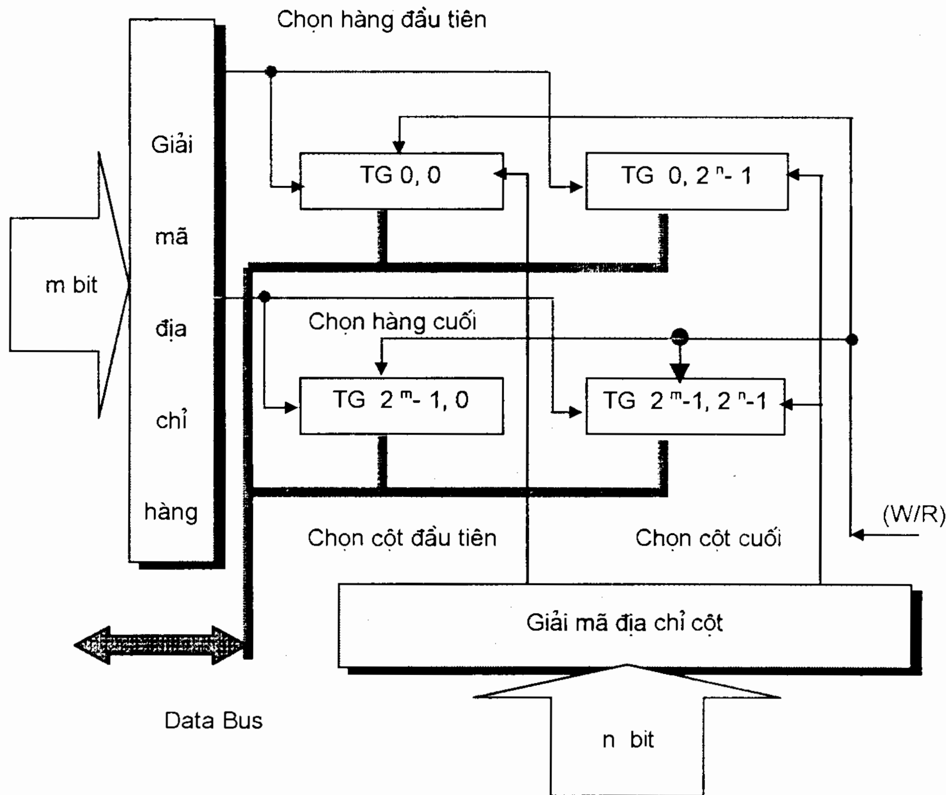
Giải.

Cho rằng các địa chỉ của cổng và các thanh ghi được khai báo ở đầu chương trình:

```
ComPort  EQU 378h;
MCR EQU 37Ch;
LSR EQU 37Dh;
MSR EQU 37Eh;
```

thì thủ tục điều khiển thu 1 byte có dạng:

```
ReceiveCOM   PROC
         PUSH AX
         PUSH DX
         MOV DX, MCR
         MOV AL, 01H ; gán DTR(bit0)= 1→thanh ghi đ.kh MODEM
         OUT DX, AL
         MOV DX, MSR
         IN AL,DX
         AND AL, 20h
         CMP AL, 20H;     trong MSR: DSR(bit5)= 1?
         JNZ  Exit_
         MOV DX, LSR
         IN AL, DX
         AND AL, 01h
         CMP AL, 01H;  trong LSR: RxDReady(bit0) 1?
         JNZ  Exit_
         MOV DX, ComPort
         IN AL, DX
         MOV Data_Receive, AL
Exit_:   POP DX
         POP AX
         IRET
ReceiveCOM   ENDP
;---------------------------------
```



***Hình 1.5.* Quản lý bộ nhớ của hệ vi xử lý**

Tương tự như vậy, bộ giải mã cột biến n dây địa chỉ ở đầu vào thành 2^n cột của ma trận quản lý. Như vậy số lượng các thanh ghi được quản lý sẽ là $2^m \times 2^n = 2^{m+n}$ thanh ghi. Dung lượng của bộ nhớ được xác định bằng số lượng bit hoặc từ (thanh ghi) thông tin mà nó có thể chứa. Nếu bộ nhớ có n+m bít (dây) địa chỉ và mỗi từ có độ dài là k bit thì dung lượng của nó sẽ là $2^{m+n} \times$ k bít. Các đơn vị của dung lượng là bit, K bit (2^{10} bit), byte (8 bit), Kbyte (2^{10} byte). Như vậy, phương pháp tổ chức địa chỉ theo ma trận hàng và cột làm giảm đáng kể số lượng dây tín hiệu trên kênh địa chỉ.

1.3.2. Bộ nhớ cố định ROM

Bộ nhớ cố định ROM (Read Only Memory) trong hệ vi xử lý dùng để lưu trữ chương trình điều hành của hệ (còn gọi là chương trình MONITOR-người hướng dẫn). Chương trình này sẽ qui định mọi hoạt động của hệ vi xử lý. Bộ vi xử lý sẽ căn cứ vào các lệnh chứa trong chương trình để điều khiển hệ vi xử lý thực hiện các chức năng, nhiệm vụ được ấn định trong lệnh. Nói cách khác, hệ vi xử lý sẽ thực hiện một cách trung thực thuật toán mà người thiết kế phần mềm đã xây dựng và cài đặt vào ROM của hệ.

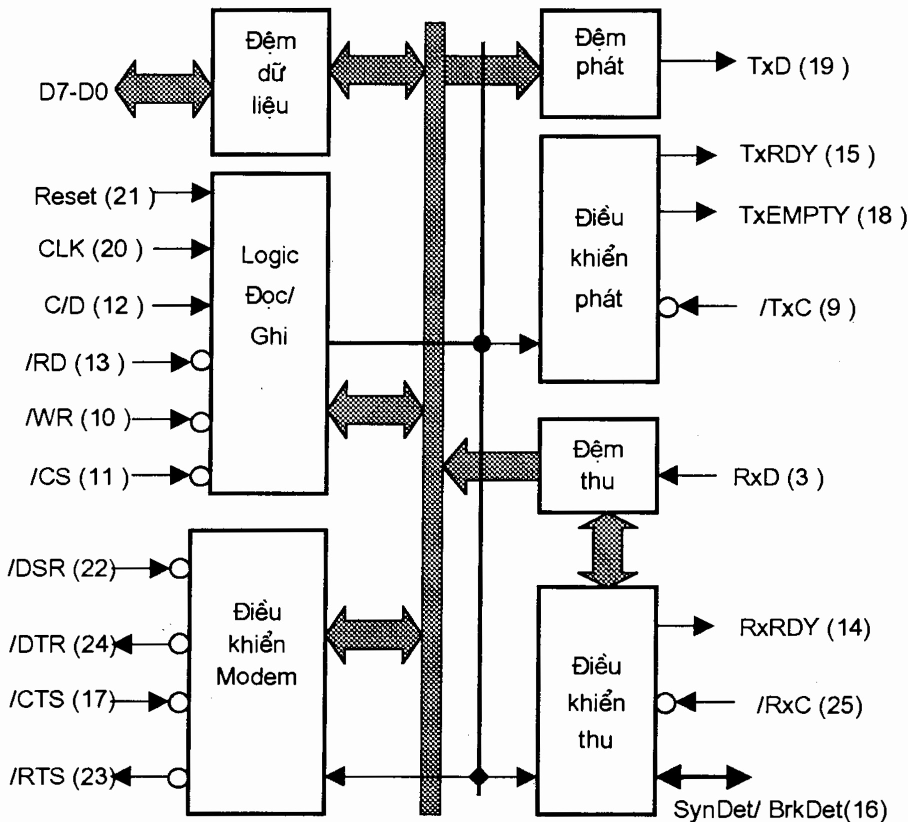
7.4. MẠCH THU PHÁT ĐỒNG BỘ VÀ DỊ BỘ VẠN NĂNG USART 8251A

7.4.1. Tổ chức của USART 8251A

USART 8251A (Universal synchronous and asynchronous receiver trasnmitter chip) có thể dùng cho cả hai kiểu truyền thông tin nối tiếp đồng bộ và dị bộ. Sơ đồ khối của mạch 8251A của Intel được biểu diễn trên hình 7.15.

Các tín hiệu của mạch 8251A hầu hết là giống các tín hiệu của bộ vi xử lý hoặc các tín hiệu tiêu chuẩn của RS 232C đã nêu trước đây. Chân chọn chip /CS của 8251A phải được nối với đầu ra của một mạch giải mã địa chỉ để đặt mạch 8251A vào một địa chỉ cơ bản nào đó. Chức năng các tín hiệu của USART gồm:

- CLK [I-hướng vào IN]: chân nối đến xung đồng bộ của hệ thống.
- TxRDY [O-hướng ra OUT]: Tín hiệu báo đệm giữ rỗng (sẵn sàng nhận ký tự mới từ CPU).
- RxRY [O]: Tín hiệu báo bộ đệm thu đầy (có ký tự nằm chờ CPU đọc vào).
- TxEMPTY [O] : tín hiệu báo cả bộ đệm giữa và bộ đệm phát đều rỗng. C/D [I]: CPU thao tác với thanh ghi lệnh/thanh ghi dữ liệu của 8251A, khi C/D = 1 thì thanh ghi lệnh được chọn làm việc. Chân này thường được nối với A0 của bus địa chỉ để cùng với các tín hiệu /WR và /RD chọn ra 4 thanh ghi bên trong 8251A.



Hình 7.15. **Sơ đồ khối của USART 8251A**

- RxC [I] và TxC [I] : Xung đồng hồ cung cấp cho các thanh ghi dịch của phần thu và phần phát. Thường 2 chân này được nối chung để phần thu và phần phát làm việc với cùng một tần số nhịp. Tần số của các xung đồng hồ đưa đến chân RxC và TxC được chọn sao cho là bội số (cụ thể là gấp 1, 16 hoặc 64 lần) của tốc độ thu hay tốc độ phát theo yêu cầu.
- SYNDET/BRKDET [I/O] : Là tín hiệu ra khi 8251A làm việc ở chế độ không đồng bộ, nếu RxD = 0 kéo dài hơn thời gian của 2 ký tự thì chân này có mức cao để báo là việc truyền hoặc đường truyền bị gián đoạn. Khi 8251A làm việc ở chế độ đồng bộ, nếu phần thu tìm thấy ký tự đồng bộ trong bản tin thu được thì chân này có mức cao. Là tín hiệu vào khi 8251A làm việc ở chế độ đồng bộ ngoài. Đó chính là nhịp đồng bộ.

Đệm ở khối phát của mạch 8251A là loại đệm kép, bao gồm bộ đệm giữ và bộ đệm phát. Trong khi 1 ký tự đang được chuyển đi từ bộ đệm phát thì một ký tự khác có thể được đưa từ CPU sang bộ đệm giữ. Các tín hiệu TxRDY và TxEMPTY sẽ cho biết trạng thái của các bộ đệm này khi mạch 8251A hoạt động.

Khi bộ đệm ở phần thu đầy thì sẽ có tín hiệu RxRDY = 1. Nếu cho đến khi phần thu của USART nhận được ký tự mới mà CPU không kịp đọc ký tự cũ thì ký tự cũ sẽ bị mất do bị đè bởi ký tự mới nhận được. Hiện tượng này gọi là hiện tượng thu đè.

7.4.2. Các thanh ghi chức năng của 8251A

Như đã nói ở trên chân C/D (giả sử nó được nối vào A0 của kênh địa chỉ) cùng các tín hiệu /WR và /RD sẽ chọn ra 4 thanh ghi bên trong của mạch USART. Đó là thanh ghi đệm dữ liệu thu, thanh ghi đệm dữ liệu phát, thanh ghi trạng thái và thanh ghi điều khiển (bảng 7.3).

Bảng 7.3

A0	/RD	/WR	Chọn ra
0	0	1	Thanh ghi đệm dữ liệu thu
0	1	0	Thanh ghi đệm dữ liệu phát
1	0	1	Thanh ghi trạng thái
1	1	0	Thanh ghi điều khiển

Với cùng một địa chỉ của thanh ghi điều khiển ta có thể thâm nhập được 2 thanh ghi khác nhau: thanh ghi chế độ và thanh ghi lệnh. Sau khi có xung Reset thì ta phải ghi liên tiếp từ chế độ rồi theo sau là từ lệnh vào địa chỉ dành cho thanh ghi điều khiển để định nghĩa các chế độ và phương thức làm việc của mạch 8251A.

Thanh ghi từ chế độ

Dạng thức của thanh ghi từ chế độ được biểu diễn như sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S2	S1	EP	PEN	L2	L1	B2	B1

Các bit D1D0 (B2B1-baud rate) là hệ số nhân của tốc độ. Giá trị của chúng quy định:

B2B1 =
- 00-đồng bộ.
- 01-x1
- 10- x 16
- 11- x 64

Các bit D3D2 (L2L1-length) quy định độ dài từ mã:

L2L1=
- 00- 5 bit.
- 01- 6 bit
- 10- 7 bit
- 11- 8 bit

Bit D4 = 1 cho phép sử dụng parity-Parity Enable.

Bit D5 = 1 parity chẵn- Even Parity.

Bit D7D6 quy định số lượng bit Stop:

S2S1 =
- 00- không hợp lệ
- 01- 1 bit
- 10- 1+1/2 bit
- 11- 2 bit

Trong từ chế độ, đối với ký tự cần truyền ta có thể chọn số bit (kiểu mã) của ký tự, số bit stop và tốc độ truyền. Nếu ta có sẵn tần số xung đồng hồ cho phần thu hoặc phần phát (giả sử là F_{clk}) và ta muốn truyền (thu/phát) dữ liệu với tốc độ *X baud*, ta phải chọn hệ số nhân tốc độ truyền *k* sao cho thoả mãn biểu thức.

F_{clk} = X.k, trong đó X là các tốc độ truyền tiêu chuẩn.

Ví dụ: Nếu ta có tần số xung đồng hồ phát là 19.200HZ và ta muốn truyền dữ liệu với tốc độ 1.200 baud thì ta phải ghi từ chế độ với 2 bit cuối là 10 để chọn được hệ số nhân tốc độ truyền là 16, vì 1200 × 16 = 19.200. Với việc dùng tần số đồng hồ cho phần thu/phát cao hơn so với tốc độ truyền ta sẽ giảm được lỗi khi truyền thông tin. Tất nhiên khi làm việc ở chế độ đồng bộ thì ta phải có từ chế độ với 2 bit cuối là 00.

Thanh ghi từ lệnh

Dạng thức của thanh ghi lệnh được biểu diễn như sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
EH	IR	RTS	ER	SBRK	RxE	DTR	TxEN

Bit D0 (Transmitt Enable) = [1-cho phép phát. / 0-cấm phát.

Bit D1 (Data Terminal Ready) = 1 sẽ làm chân ra /DTR=0 (tích cực)

Bit D2 (Receiver Enable) = [1-cho phép thu. / 0-cấm thu.

Phát ký tự báo cắt (Break) = [1- làm chân ra /TxD=0 (cắt). / 0 - phát bình thường.

Bit D4 = 1 -xoá các cờ báo lỗi PE, OE, FE

Bit D5 (Request to send) yều cầu truyền = 1 làm chân /RTS = 0.

Bit D6 = 1- Khởi động mềm.

Bit D7 = 1-Tìm ký tự đồng bộ.

Từ lệnh phải được ghi vào 8251A ngay sau khi ghi từ chế độ. Mọi từ ghi vào 8251A sau khi ghi từ chế độ đều được coi là từ lệnh. Mỗi bit của từ lệnh nang một ý nghĩa riêng.

Bit D0 = 1 cho phép phần phát làm việc, nếu lúc này ta đã có /CTS = 0 thì bắt buộc chân TxRDY = 1 để báo cho CPU là đệm giữ rỗng. Tín hiệu này có thể được nối với chân INTR của CPU để lưu ý CPU đưa ký tự sang 8251A theo cách điều khiển trao đổi dữ liệu bằng ngắt.

Bit D1 = 1 cho phép điều khiển tín hiệu có mức thấp tại chân /DTR.

Bit D2 = 1 cho phép phần thu làm việc, nếu lúc này ta có ký tự tại đệm thu (sẵn sàng để CPU đọc) thì bắt buộc chân RxRDY=1, báo cho CPU là đệm thu đầy. Tín hiệu này có thể được nối với chân INTR của CPU để lưu ý CPU đọc ký tự từ 8251A theo cách điều khiển trao đổi dữ liệu bằng ngắt.

Bit D3 = 1 cho phép 8251A đưa ra ký tự gián đoạn (ký tự với tất cả các bit bằng 0) tại chân TxD.

Bit D4 = 1 cho phép xoá tất cả các bit cờ trong thanh ghi từ trạng thái của 8251A gồm cờ lỗi parity (PE), cờ thu đè (OE) và cờ lỗi khung (FE) về 0.

Bit D5 = 1 cho phép điều khiển có mức thấp tín hiệu tại chân /RTS, (/RTS=0).

Bit D6 = 1 cho phép xoá mạch 8251A bằng chương trình về trạng thái nhận lệnh ban đầu. Sử dụng khả năng này để đảm bảo chắc chắn là USART 8251A bị xoá. Lúc này lại phải ghi từ chế độ và sau đó là từ lệnh cho 8251A.

Trong chế độ đồng bộ, bit D7 = 1 cho phép mạch 8251A bắt đầu tìm ký tự đồng bộ trong bản tin thu được. Nếu tìm được, nó sẽ đưa thông báo ra bằng mức cao trên chân SYNDET/BRKDET.

USART cho phép đọc được các thông tin trạng thái làm việc của mạch 8251A bằng cách đọc vào CPU nội dung các thanh ghi trạng thái của nó. Nhờ đọc các bit trạng thái ta biết được tình trạng của việc truyền dữ liệu và ta có thể dùng các bit này vào để viết chương trình điều khiển bằng phương pháp thăm dò việc trao đổi dữ liệu giữa CPU và USART.

Thanh ghi trạng thái có dạng thức sau:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
DSR	SYNDET	FE	OE	PE	TxEMPTY	RxRDY	TxRDY

Bit D0 = 1: phần phát sẵn sàng.

Bit D1 = 1: phần thu sẵn sàng.

Bit D2 = 1: đệm giữ và đệm phát rỗng.

Bit D3D4D5 = 111: lỗi Parity, lỗi Overrun, lỗi Frame.

Bit D6 = 1: đã tìm được ký tự đồng bộ.

Bit D7 = 1: Modem sẵn sàng.

Ý nghĩa của các bit trạng thái đã khá rõ ràng. Cần nói thêm là lỗi khung chỉ có ý nghĩa khi truyền không đồng bộ. Để điều khiển trao đổi dữ liệu kiểu thăm dò ta sẽ phải liên tục đọc và kiểm tra các bit trạng thái liên quan để có quyết định cụ thể.

7.4.3. Nối ghép USART 8251A với hệ vi xử lý

Căn cứ vào sơ đồ hình 7.16 ta thấy kênh dữ liệu của USART được nối với hệ vi xử lý thông qua chip vào/ra 245. Hướng truyền tin của nó được điều khiển bằng tổ hợp /IORD, /IOWR của hệ vi xử lý và /CS của USART 8251A. Các chân tín hiệu thu phát và giao tiếp với Modem hoàn toàn sáng tỏ và ghép nối chúng dễ dàng do các bên đều có tín hiệu có tên và chức năng tương ứng. Tín hiệu chip select /CS được tổ chức trên chip 138 nên dễ dàng quy định được địa chỉ cơ sở cho phù hợp. Riêng bộ tạo giao động được lấy từ hệ vi xử lý sang và nối với đầu vào CLK của USART. Các tần số thu phát TxC và RxD được nối chung và lấy từ chân ra OUT1 của bộ chia tần có lập trình Timer 8253. Tần số vào của 8253 cũng chính là tần số CLK của USART.

***Hình 7.16.* Ghép nối USART với hệ vi xử lý**

Khi tổ chức ghép nối xong cần khởi tạo cho USART 8251A. Theo sơ đồ trên thì 8251A có địa chỉ cơ bản 3F8h và chân C/D nối với bit A0 của kênh địa chỉ của hệ vi xử lý. Lưu ý rằng muốn cho việc lập trình để khởi đầu cho 8251A được tin cậy, thay vì sử dụng tín hiệu RESET cứng ta phải gán được trạng thái ban đầu cho 8251A bằng các lệnh ghi giá trị 00h vào thanh ghi điều khiển một số lần.

Khi thực hiện ghi liên tiếp vào USART phải đảm bảo không nhanh quá thời gian hồi phục $T_{rv} = 16xT_{clk}$ giữa 2 lần ghi, tức là ta phải có trễ giữa hai lần ghi liên tiếp vào USART 8251A để mạch này hoạt động được tin cậy. Chương trình minh hoạ dưới đây thể hiện thuật toán này.

```
CODE_SEG SEGMENT
ASSUME CS:CODE_SEG,DS:CODE_SEG
        ORG 100H
;----------------------------------------------------------------------
```

```
;Địa chỉ cơ sở của USART = 3F8h
        USART0 EQU 3F8h;  A0=0=C/D (TG đệm thu, đệm phát)
        USART1 EQU 3F9h;  A0=1=C/D(TG trạng thái, điều khiển)
;--------------------------------------------------------------------------
MAIN PROC
 CALL INIT
  ; Tại đây là các lệnh của chương trình chính
MAIN ENDP
;--------------------------------------------------------------------------
INIT PROC
        PUSH AX
        PUSH CX
        PUSH DX
        MOV DX,USART1; địa chỉ TG điều khiển → DX
        MOV AL,00H  ; Al←00h
        MOV CX,3
LOOP_3_TIMES:
        OUT DX,AL
        CALL DELAY; gọi chương trình con tạo trễ
        LOOP  LOOP_3_TIMES ; quay lại để nạp lại
        MOV AL,40H;  AL=0100 0000b – lệnh xoá mềm (bit D6=1)
        OUT DX,AL; đưa ra TG lệnh
        CALL DELAY
        MOV AL,7EH ; đặt chế độ không đồng bộ với:
        OUT DX,AL    ; 1 bit STOP, PRITY chẵn, 8 bit dữ liệi, x16
        CALL DELAY
        MOV AL, 37H ; từ lệnh 0011 0111b là: RTS=1, ER=1,RxE=1,
        OUT DX,AL    ; DTR=1, TxEn=1
        CALL DELAY
        POP DX
        POP CX
        POP AX
        RET
```

```
 INIT ENDP
;--------------------------------------------
DELAY PROC
        PUSH CX
        MOV CX,10
LOOP_DELAY:     LOOP LOOP_DELAY
        POP CX
        RET
 DELAY ENDP
;--------------------------------------------
CODE_SEG ENDS
END MAIN
```

Đoạn chương trình với vòng lặp tại nhãn LOOP_3_TIMES để ghi 3 lần giá trị 00h (mỗi lần có trễ dài hơn $16T_{clk}$) vào thanh ghi điều khiển của USART 8251A. Tiếp theo là lệnh xoá mềm USART 8251A để chuẩn bị nạp từ chế độ mới. Sau đó mới đưa từ chế độ và tiếp theo là từ lệnh vào để khởi đầu cho 8251A. Giữa các lệnh ghi vào 8251A là các vòng lặp để tạo thời gian trễ cần thiết.

7.5. TỔ CHỨC HỆ THỐNG TRUYỀN SỐ LIỆU

Hình 7.17. **Hệ vi xử lý dùng USART 8251A và PIC 8259A**

Hệ thống truyền số liệu có thể tổ chức như sơ đồ 7.17 trên chip UART 8251A. Cơ chế ngắt được thực hiện nhờ chip PIC 8259A. Ngắt thu sử dụng đầu ngắt IR0 của PIC để đưa số hiệu ngắt vào hệ vi xử lý. Ngắt phát sử dụng đầu ngắt IR1 của PIC để đưa số hiệu ngắt vào hệ vi xử lý.

Bài tập: Căn cứ vào sơ đồ 3.17 xây dựng chương trình con phục vụ ngắt thu IR0_THU và chương trình con phục vụ ngắt phát IR1_PHAT để thu phát từng byte. Biết rằng vector ngắt cơ sở của PIC là 70H. Địa chỉ chọn chip /CS của PIC là 00h còn địa chỉ chọn chip /CS của UART là 3F8h. Dữ liệu phát chứa trong vùng nhớ có địa chỉ OFFSET là 700h. Dữ liệu thu được phải lưu vào bộ nhớ của chương trình.

Giải: Căn cứ vào yêu cầu đầu bài và sơ đồ chức năng của hệ, ta có chương trình sau:

```
;*************************************************************
  PIC0 EQU 0000H;  a0=0(icw1, ocw2, ocw3)
  PIC1 EQU 0001H;  a0=1(icw2, icw3, icw4, ocw1)
  USART0      EQU  3F8H ;A0=0 (bộ đệm thu,phát)
  USART1      EQU  3F9H ;A0=1 (TG điều khiển/trạng thái)
  BUFFER_PHAT      EQU  700H
;*************************************************************
CODE_SEG    SEGMENT
ASSUME      CS:CODE_SEG, DS:DATA_SEG
 ORG    100H
;*************************************************************
MAIN  PROC
      MOV  AX,0000H   ;cài vector ngắt
      MOV  ES,AX
      MOV  DX,OFFSET IR0_THU        ; Vector ngắt cho CTCPVN Thu
      MOV  WORD PTR ES:[70H*4],DX
      MOV  WORD PTR ES:[70H*4+2],CS
      MOV  DX,OFFSET IR1_PHAT     ; Vector ngắt cho CTCPVN Phát
      MOV  WORD PTR ES:[71H*4],DX
      MOV  WORD PTR ES:[71H*4+2],CS
      MOV  BX, OFFSET BUFFER_THU ; tổ chức con trỏ
      MOV  BUFFER_THU_PTR, BX
      MOV  BX, OFFSET BUFFER_PHAT
```

```
        MOV  BUFFER_PHAT_PTR, BX
        CALL INIT
    ;------------------------------------------
    ;các lệnh khác của MAIN PROC viết tại đây
    ;------------------------------------------
MAIN    ENDP
;********************************
IR0_THU      PROC
        PUSH AX
        PUSH BX
        PUSH DX
        IN AL, PIC1          ;lấy nội dung của mặt nạ ngắt 8259
        MOV  AH,AL           ;Cất vào AH
        MOV  AL,0FEH         ;Cho ngắt IR0 hoạt động
        OUT  PIC1,AL
        MOV  DX, USART0;bộ đệm thu
        IN   AL,DX           ;Thu thông tin
        MOV  BYTE PTR [BUFFER_THU_PTR], AL;lưu vào bộ nhớ thu
        INC  BUFFER_THU_PTR;
        MOV  AL,AH                     ;khôi phục mặt nạ ngắt
        OUT  PIC1,AL
        MOV  AL,20H                    ;mã EOI cho PIC 8259
        OUT  PIC0,AL
        POP  DX
        POP  CX
        POP  AX
        IRET
IR0_THU      ENDP
;********************************
IR1_PHAT     PROC
        PUSH AX
        PUSH BX
        PUSH DX
```

```
        IN AL,PIC1
        MOV  AH,AL
        MOV  AL,0FCH
        OUT  PIC1,AL
        MOV  DX,USART0 ; bộ đệm phát
        MOV  AL, BYTE PTR [ BUFFER_PHAT_PTR]
        OUT  DX,AL ;lưu vào bộ nhớ do BUFFER_PHAT_PTR trỏ tới
        INC BUFFER_PHAT_PTR
        MOV  AL,AH                  ;khôi phục mặt nạ ngắt
        OUT  PIC1,AL
        MOV  AL,20H
        OUT  PIC0,AL
        POP  DX
        POP  CX
        POP  AX
        IRET
IR1_PHAT     ENDP
;********************************
INIT    PROC        ; khởi tạo USART 8251
        PUSH AX
        PUSH CX
        PUSH DX
        CLI  ; cấm mọi ngắt
        MOV  DX, USART1
        MOV  AL,00          ;Xóa TG điều khiển và TG lệnh
        MOV  CX,3
 LOOP_3_TIMES:
        OUT  DX,AL
        CALL DELAY                  ;giữ chậm
        LOOP LOOP_3_TIMES
        MOV  AL,40H                 ;
        OUT  DX,AL                  ;Khởi động mềm 01000000B
        CALL DELAY                  ;(giữ chậm)
```

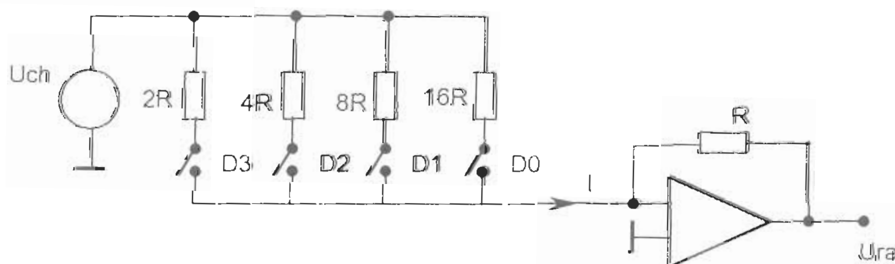
```
DELAY ENDP
;**********************************
CODE_SEG    ENDS
;**********************************
DATA_SEG    SEGMENT
  BUFFER_THU DB  1024  DUP(0) ;cấp phát 1024 byte cho bộ đệm thu
  BUFFER_THU_PTR        DW(0) ; con trỏ bộ đệm thu
  BUFFER_PHAT_PTR       DW(0)  ; con trỏ bộ đệm phát
DATA_SEG    ENDS
;**********************************
END    MAIN
```

CHƯƠNG 8
BIẾN ĐỔI TÍN HIỆU TƯƠNG TỰ - SỐ VÀ TÍN HIỆU SỐ - TƯƠNG TỰ

Các tín hiệu trong môi trường tự nhiên phần lớn là các tín hiệu tương tự, nghĩa là các tham số cơ bản của nó là hàm liên tục của thời gian. Mặt khác, vấn đề trao đổi thông tin với bên ngoài lại là nhiệm vụ trọng yếu của hệ vi xử lý, nên để có thể giao tiếp được với môi trường bên ngoài hệ phải được trang bị khả năng biến đổi tín hiệu từ tương tự sang số khi nhận vào và từ tín hiệu số sang tương tự khi xuất thông tin ra. Các bộ biến đổi đó được gọi là bộ biến đổi số-tương tự DAC (Digital To Analog Converter) và bộ biến đổi tương tự-số ADC (Analog To Digital Converter). Chương này sẽ nghiên cứu nguyên tắc làm việc và phương pháp ghép nối chúng với hệ vi xử lý.

8.1. NGUYÊN TẮC HOẠT ĐỘNG CỦA BỘ BIẾN ĐỔI SỐ - TƯƠNG TỰ

Trong kỹ thuật xử lý tín hiệu bằng kỹ thuật vi xử lý, có rất nhiều thiết bị ngoại vi chỉ có thể làm việc với tín hiệu tương tự nên hệ buộc phải thực hiện phép biến đổi tín hiệu cho phù hợp. Để thực hiện nhiệm vụ biến đổi tín hiệu từ dạng tín hiệu số về dạng tín hiệu tương tự cần sử dụng bộ biến đổi DAC. Nguyên lý chung của bộ biến đổi DAC có thể được xây dựng theo các phương pháp khác nhau nhưng phương pháp lấy tổng dòng trên các điện trở trọng số là phổ biến hơn cả. Với phương pháp này nguyên tắc biến đổi của nó dựa trên một mạch khuyếch đại có điện trở phản hồi hoạt động dưới dạng một mạch cộng dòng như hình 8.1 đã chỉ ra.



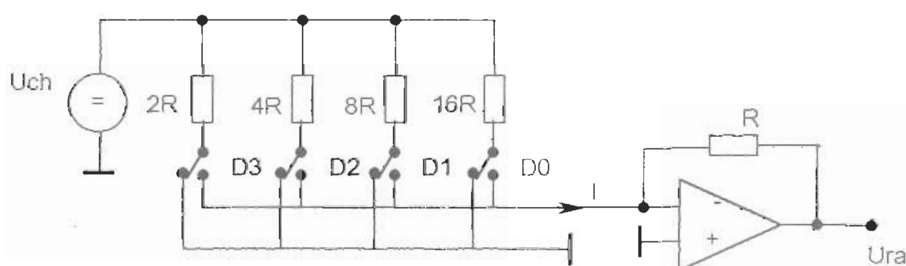
Hình 8.1. Sơ đồ mạch DAC theo nguyên lý mạch cộng dòng

Bộ biến đổi DAC có 3 thành phần chủ yếu là nguồn điện áp chuẩn Uch, đóng vai trò thước đo đơn vị biến đổi nên độ chính xác của nó có ảnh hưởng lớn tới độ chính xác của cả mạch. Thành phần thứ hai là các điện trở trọng số để tạo dòng điện theo hệ nhị phân qua các khoá chuyển mạch điện tử. Khi các khoá điện tử trong sơ đồ hình 8.1 là D0, D1, D2, D3 sẽ đóng mở tùy thuộc trạng thái của số đầu vào thì ta sẽ có hệ thức:

$$I = \frac{-U_{ch}}{R} . \left(2^{-1}D_3 + 2^{-2}D_2 + 2^{-3}D_1 + 2^{-4}D_0\right)$$

Như vậy dòng ra sẽ tương ứng với mã số nhị phân dòng vào. Thành phần thứ ba chính là mạch khuếch đại thuật toán, bảo đảm hệ số khuếch đại lớn và tuyến tính.

Tuy nhiên phương pháp biến đổi trên có biên độ điện áp lớn đặt vào mạch khuyếch đại thuật toán, vì vậy người ta thường sử dụng phương pháp dùng khoá đổi chiều để khắc phục khó khăn trên. Sơ đồ thực hiện theo phương pháp này có dạng như hình 8.2.

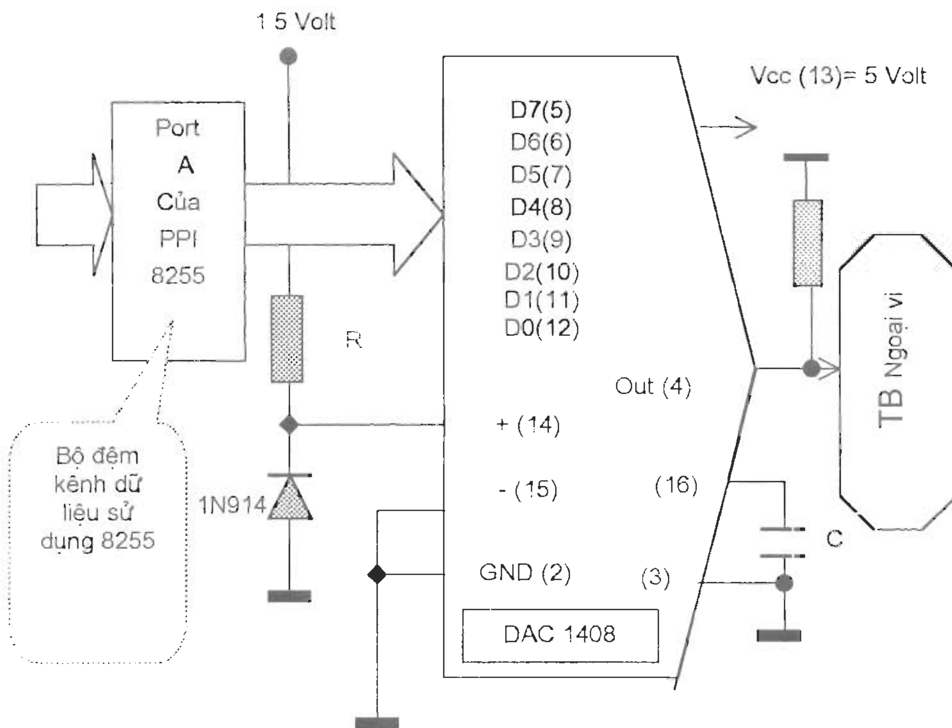


Hình 8.2. Sơ đồ mạch DAC theo nguyên lý dùng khoá đổi chiều

Với phương pháp này, dòng qua mỗi điện trở là không đổi do đó nguồn điện áp chuẩn là cố định. Điện áp đầu ra của bộ DAC là dạng tín hiệu có biên độ tỷ lệ với bậc của mã số nhị phân đầu vào.

Dòng DAC có dạng nấc thang, mỗi nấc thang sẽ tương ứng với một số nhị phân bốn bít của lối vào (bộ DAC 4 bít). Khi đặt giá trị cực đại thì DAC sẽ trở về giá trị ban đầu và chu kỳ tiếp theo sẽ lặp lại đúng 16 nấc như vậy. Mỗi nấc điện áp được gọi là 1 LSB vì sự thay đổi của chúng ứng với sự chuyển trạng thái của bit có trọng số thấp nhất.

DAC 1408 (tương đương 0808) là bộ biến đổi số-tương tự 8 bit thông dụng mà nguyên lý làm việc của nó hoàn toàn tương tự như đã trình bày ở trên. Khi sử dụng DAC 1408 với hệ vi xử lý chỉ cần ghép theo sơ đồ hình 8.3 là có thể thực hiện được phép biến đổi từ các mã số nhị phân bên trong hệ thành các mức điện áp hay dòng điện tương ứng cho thiết bị ngoại vi.

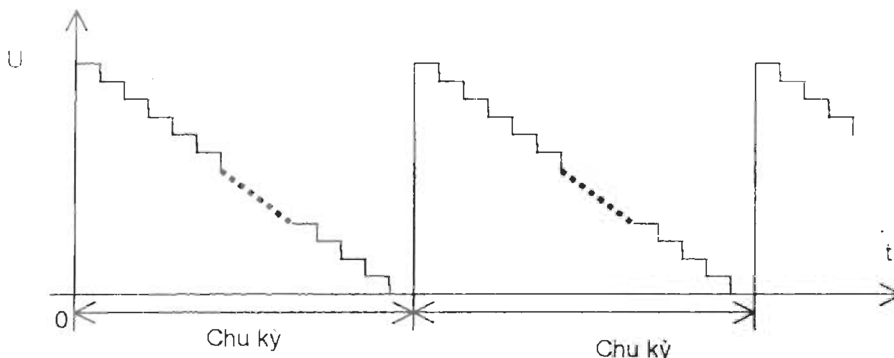


Hình 8.3. **Ghép nối Bộ biến đổi DAC 1408 với hệ vi xử lý**

Đoạn chương trình sau đây minh hoạ khả năng làm việc của DAC 1408 được điều khiển từ phía hệ vi xử lý:

```
        MOV DX, PORT_A_8255;     DX= địa chỉ cổngPortA của 8255
        MOV AL, 0FFH; Gán giá trị khởi đầu cho AL=max
LOOP_Count:
        OUT DX, AL; đưa ra cho DAC 1408 để thực hiện biến đổi
        DEC AL  ; giảm AL xuống 1 đơn vị
        JNZ LOOP_Count    ;lặp lại
```

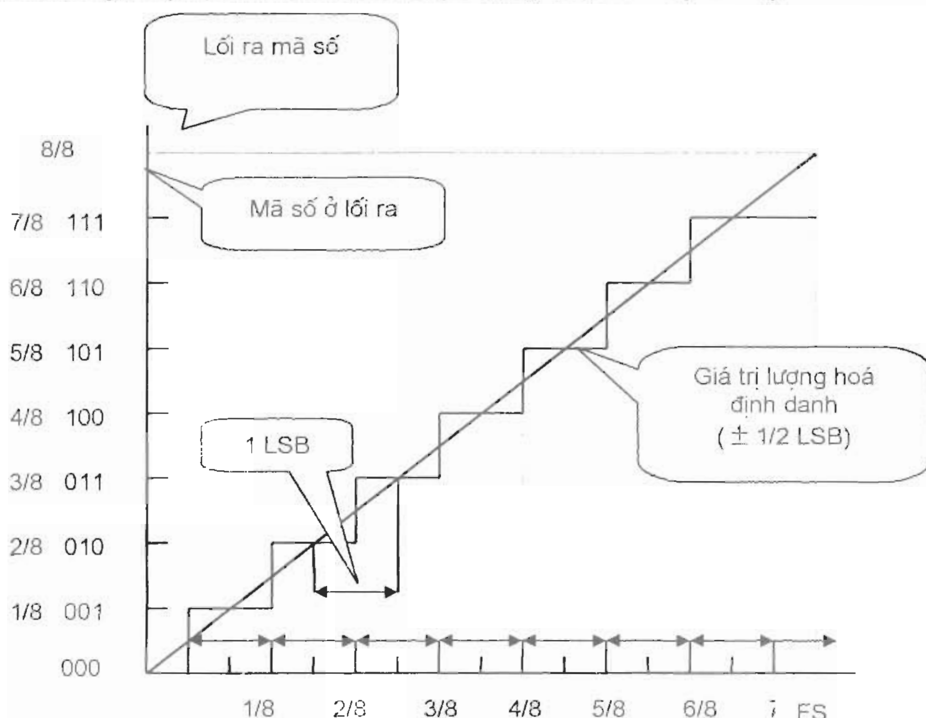
Đồ thị thời gian của tín hiệu ra sẽ là chu kỳ dạng bậc thang 256 mức điện áp từ giá trị cực đại tới giá trị cực tiểu rồi lặp lại.



8.2. NGUYÊN TẮC HOẠT ĐỘNG CỦA BỘ BIẾN ĐỔI TƯƠNG TỰ -SỐ

Bộ biến đổi tương tự -số ADC đóng vai trò quan trọng trong hệ thống xử lý thông tin khi mà các luồng tín hiệu đưa vào hệ vi xử lý là tín hiệu dạng tương tự.

Các bộ chuyển đổi ADC thực hiện hai chức năng cơ bản là lượng tử hoá và mã hoá. Lượng tử hoá là gán những giá trị của một tín hiệu tương tự vào vùng các giá trị rời rạc có thể xảy ra trong quá trình lượng tử hoá. Mã hoá là gán những giá trị nhị phân cho từng giá trị rời rạc sinh ra trong quá trình lượng tử hoá. Đối với ADC ta cũng dùng các loại mã số như nhị phân, BCD, bù hai, bù một. Hình 8.4 cho biết đặc tuyến của một ADC 3 bít làm việc với mã nhị phân tự nhiên. Một ADC n bít có 2^n tổ hợp mã ra khác nhau, như vậy ADC 3 bít có 8 mã ra khác nhau, chúng được biểu diễn trên trục tung của đồ thị thời gian.



Hình 8.4. **Đồ thị biến đổi của ADC 3 bit (FS- Full Scale)**

Trên trục hoành biểu diễn giá trị của điện áp vào tương tự. Độ lớn của mỗi đơn vị lấy mẫu do phép lượng tử hoá quy định là: $Q = \frac{FS}{2^n}$. Điểm giữa mỗi "mẫu" là giá trị điện áp tương tự được biểu diễn bằng một mã nhị phân ra tương ứng với mẫu đó. Thí dụ giữa $\frac{1}{16}$FS và $\frac{3}{16}$FS là điểm $\frac{1}{8}$FS, giá trị điện áp vào tương tự trong khoảng từ $\frac{1}{16}$FS đến $\frac{3}{16}$FS được chuyển đổi sang mã số là 001 ứng

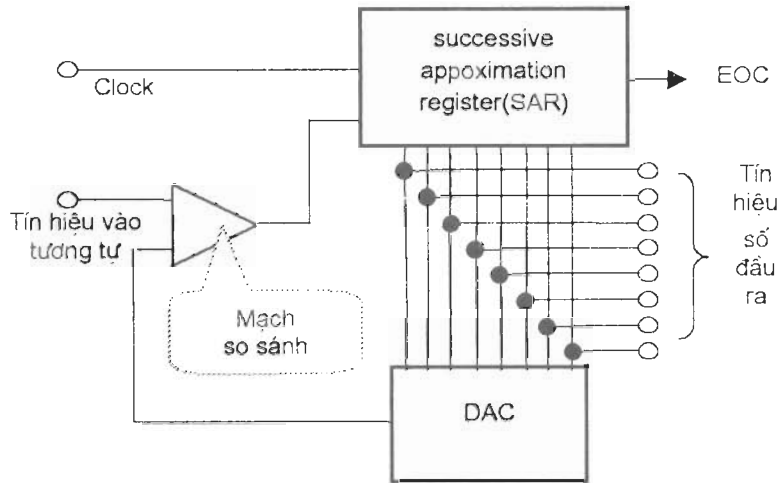
với giá trị điện áp vào là $\frac{1}{8}$ FS. Như vậy phép lượng tử hoá tự nó đã bao hàm sai số là $\pm \frac{Q}{2}$ ở trường hợp lý tưởng, giá trị M ở lối ra cho biết một giá trị là $M \pm \frac{Q}{2}$ ở lối vào ($M \pm \frac{FS}{2^{n+1}}$). Như vậy chỉ có thể giảm sai số này bằng cách tăng số bit cho ADC. Trong thực tiễn vì có những sai số như sai số độ lệch, sai số phi tuyến nên việc xác định những điểm giá trị của điện áp vào không chính xác. Đồng thời do sai số toàn bộ của một ADC bao gồm cả sai số lượng tử hoá nên không thể tăng số bit lên quá nhiều, tới mức mà sai số lượng tử hoá có thể so sánh được với những sai số kể trên.

Một ADC n bít được xây dựng theo một trong hai phương pháp là phương pháp trực tiếp và phương pháp gián tiếp. Trong phương pháp trực tiếp, điện áp tương tự cần chuyển đổi được so sánh liên tục với điện áp ra của một DAC khi mã nhị phân ở lối vào của nó liên tục thay đổi. Khi có sự cân bằng giữa hai điện áp này, mã nhị phân ở lối vào của DAC bấy giờ chính là mã kết quả. Trong phương pháp chuyển đổi gián tiếp, điện áp cần chuyển đổi trước hết được chuyển đổi sang một đại lượng trung gian, sau đó đại lượng này mới được chuyển đổi sang mã số. Phương pháp này nói chung có tốc độ chuyển đổi chậm hơn nhiều so với phương pháp trực tiếp. Vì vậy phương pháp chuyển đổi trực tiếp được sử dụng phổ biến. Có hai phương pháp chuyển đổi trực tiếp là phương pháp ADC có đếm (counting ADC) và phương pháp ADC xấp xỉ liên tiếp.

Trong thực tế thường sử dụng loại ADC xấp xỉ liên tiếp. Trong phương pháp này có một ưu điểm lớn là thời gian chuyển đổi chỉ tỷ lệ thuận với số lượng bit của mã số và thời gian thiết lập của thanh ghi xấp xỉ liên tiếp chứ không phụ thuộc vào độ lớn của điện áp cần chuyển đổi. Để thực hiện quá trình chuyển đổi người ta cần đặt lần lượt mỗi bit của mã số lên một, bắt đầu từ bit cao nhất (MSB). Sơ đồ mô tả ADC kiểu này được trình bầy ở hình 8.5. Thanh ghi SAR (thanh ghi xấp xỉ liên tiếp) điều khiển lối vào của DAC theo thuật toán sau:

1. Đặt bit cao nhất (MSB) bằng 1.
2. Nếu lối ra của mạch so sánh là 1 thì bit này bị xoá về 0, nếu không phải thì giá trị đó được giữ nguyên.
3. Bit cao nhất tiếp theo được đặt lên 1 rồi lặp lại bước 2. Nếu tất cả các bit đã được xét thì quá trình chuyển đổi thực hiện xong.

Thuật toán này trước hết kiểm tra xem điện áp cần chuyển đổi lớn hay nhỏ hơn FS/2. Nếu là lớn hơn, khi MSB = 1 ta có điện áp là logic 0, ta giữ nguyên MSB =1 và xét tiếp bit cao nhất tiếp theo xem điện áp vào lớn hơn hay nhỏ hơn 3/4 FS... Cứ như vậy ta xét đến bit cuối cùng (LSB), lúc đó nội dung của thanh ghi SAR sẽ là mã số kết quả. Đồng thời tại thời điểm này một đầu ra của SAR đưa ra một tín hiệu EOC (End of Converter) báo đã chuyển đổi xong.



Hình 8.5. Sơ đồ một ADC theo phương pháp xấp sỉ liên tiếp

Thanh ghi xấp xỉ liên tiếp được tổ chức dưới dạng vi mạch với tất cả những phần tử điều khiển cần thiết để có thể xây dựng những ADC với tốc độ cao. Trong thực tế có những thanh ghi xấp xỉ liên tiếp với cấu trúc theo những mã số sử dụng khác nhau.

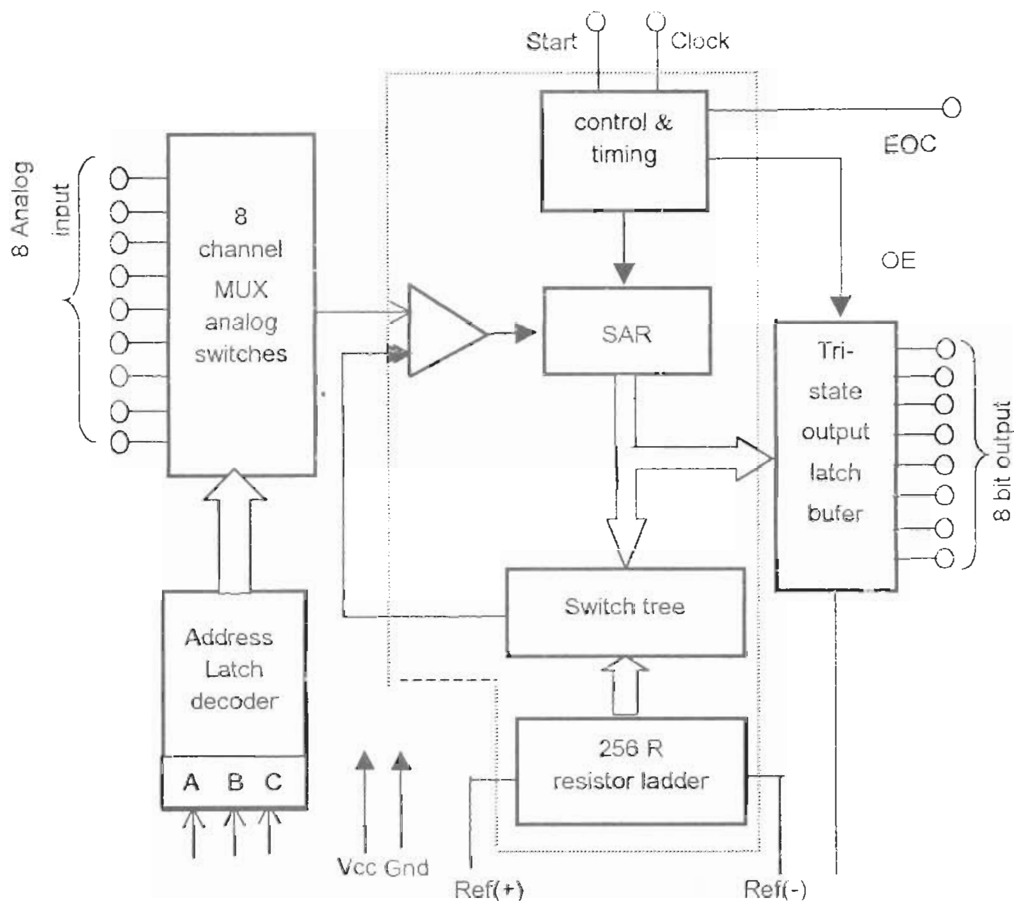
8.3. BỘ BIẾN ĐỔI ADC 8 BIT 0809

Bộ biến đổi ADC 0809 là ADC thông dụng được sử dụng rất rộng rãi có tám đầu vào tương tự và đầu ra 8 bit số, chuyển đổi theo phương pháp xấp xỉ liên tiếp. Các đặc trưng cơ bản của ADC 0809 là:

- Đầu ra có bộ đệm ba trạng thái để ghép trực tiếp vào kênh dữ liệu của hệ Vi xử lý .
- Dải tín hiệu lối vào tương tự 5v khi nguồn nuôi là +5V. Có thể mở rộng thang đo bằng các giải pháp kỹ thuật cho từng mạch cụ thể.
- Không đòi hỏi điều chỉnh "0".
- Thời gian biến đổi 100µs.
- Sai số tổng cộng ± 1/2 LSB
- Sử dụng nguồn nuôi đơn +5V, hiệu suất cao.
- Đảm bảo sai số tuyến tính trong dải nhiệt độ từ -40 đến +85^{0}C .

8.3.1. Sơ đồ chức năng của ADC 0809

Sơ đồ khối của bộ chuyển đổi ADC 0809 trên hình 8.6.



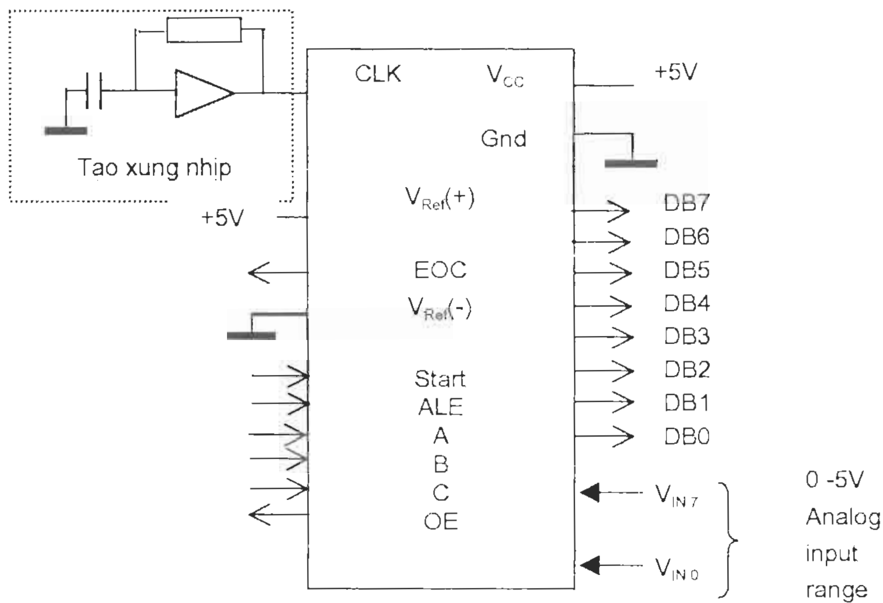
Hình 8.6. Sơ đồ khối bộ chuyển đổi ADC 0809

8.3.2. Ghép tín hiệu vào ADC 0809

Trong chu trình chuyển đổi dòng đầu vào, tín hiệu giữ nhịp 500 KHz dùng cho bộ ADC 0809 được tạo ra ở bên ngoài và được đưa đến chân Clock (Bộ chuyển đổi ADC 0809 sử dụng tần số Clock có thể trong dải 200KHz ÷ 1MHz).

Ghép tín hiệu đơn cực đến đầu vào của ADC 0809

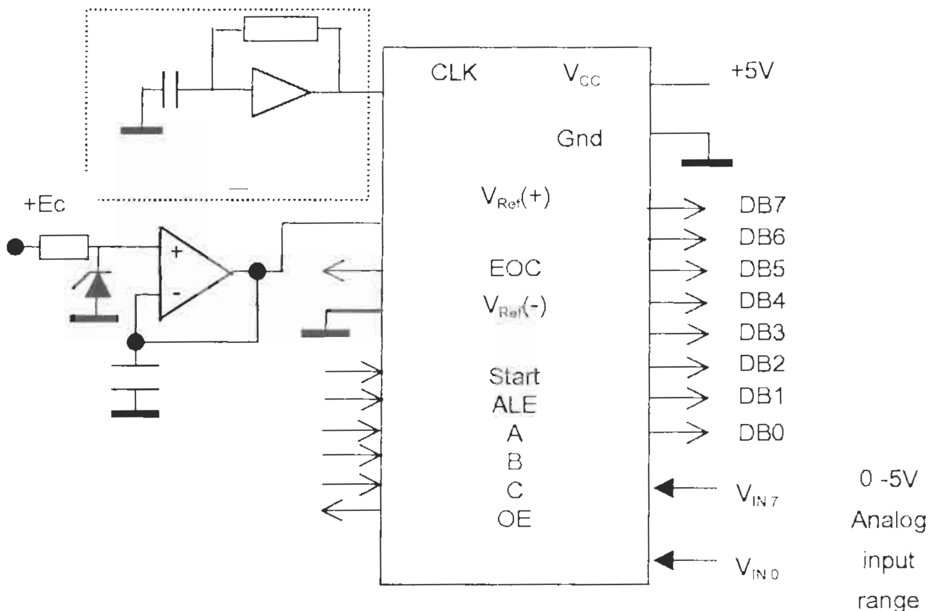
Khi tín hiệu đầu vào bộ chuyển đổi ADC là đơn cực (một cực tính), sơ đồ nguyên lý khi đó được mắc như hình 8.7. Trong trường hợp này thang đo tương ứng 5V cho các đầu vào (tức là các đại lượng cần đo qua các bộ cảm biến phải đưa về dải từ 0V đến 5V. Tương ứng với mức 0V ở đầu vào là tổ hợp nhị phân 0000 0000 ở đầu ra. Còn tương ứng với mức 5V có tổ hợp 1111 1111 ở đầu ra.



Hình 8.7. **Hoạt động của ADC 0809 khi đầu vào là đơn cực**

Ghép tín hiệu lưỡng cực đến đầu vào của ADC 0809

Trong thực tế khảo sát các đại lượng cần biến đổi thường là các tín hiệu xoay chiều (tín hiệu vào có hai cực tính). Với trường hợp này cần mở rộng thang đo thích hợp với tín hiệu. Sơ đồ nguyên lý của mạch thể hiện dạng này như hình 8.8.



Hình 8.8. **Hoạt động của ADC 0809 khi đầu vào là lưỡng cực**

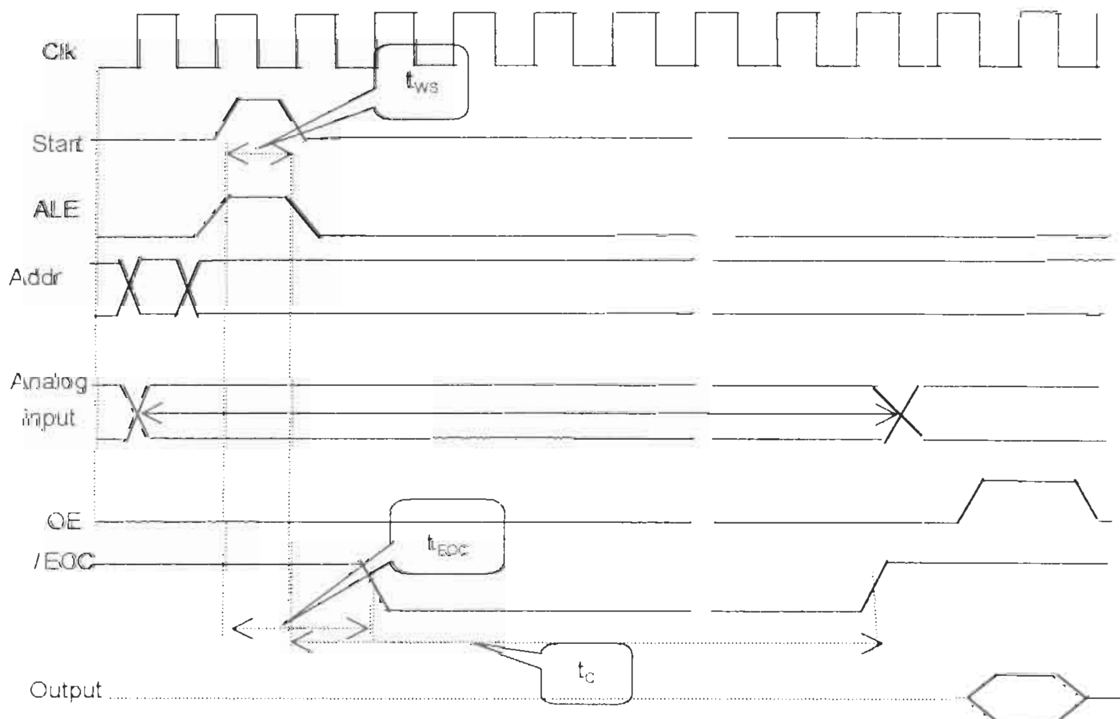
Để mở rộng thang đo, điện áp so sánh được đưa qua tầng lặp lại điện áp để đưa đến chân 12 (chânRef (+)) của ADC 0809. Các kênh lối vào tương tự được dẫn đến các chân V_{IN0} đến V_{IN7}.

Nếu chọn $V_{Ref(+)}$ = +2,5V thì từ đầu ra của các bộ cảm biến phải đưa về dải 0 đến 2,5 V (cho phép đo trong dải ± 2,5V). Khi đó cận trên của thang đo (+2,49V) tương ứng với tổ hợp mã nhị phân ở đầu ra là 1111 1111. Cận dưới thang đo (-2,49V) ứng với tổ hợp mã nhị phân ở đầu ra là 0000 0000. Còn mức "0" của thang đo ứng với tổ hợp : 0111 1111.

Đồ thị thời gian của ADC 0809

Các bit địa chỉ ở lối vào A,B,C từ bộ giải mã địa chỉ sẽ chốt và xác định kênh đầu vào nào được lựa chọn. Tương ứng với kênh đầu vào xác định ở trên khi có một xung dương đặt vào chân số 6 (Start) với độ rộng tối thiểu (t_{WS} = 200ns), sau thời gian t_{EOC} (tính từ sườn trước của xung Start) bộ chuyển đổi bắt đầu thực hiện việc chuyển đổi và trong suốt thời gian này chân tín hiệu ra EOC luôn ở mức thấp. Đồng thời đầu ra ba trạng thái của ADC 0809 bị thả nổi (ở trạng thái trở kháng cao).

Sau thời gian t_C = 100μs, ADC 0809 thực hiện chuyển đổi xong, dữ liệu đầu vào được đưa đến bộ đệm đầu ra ba trạng thái, đồng thời chân tín hiệu EOC chuyển lên mức cao báo hiệu cho hệ VXL biết để đọc kết quả vào. Hoạt động của mạch được thể hiện trên đồ thị hình 8.9.



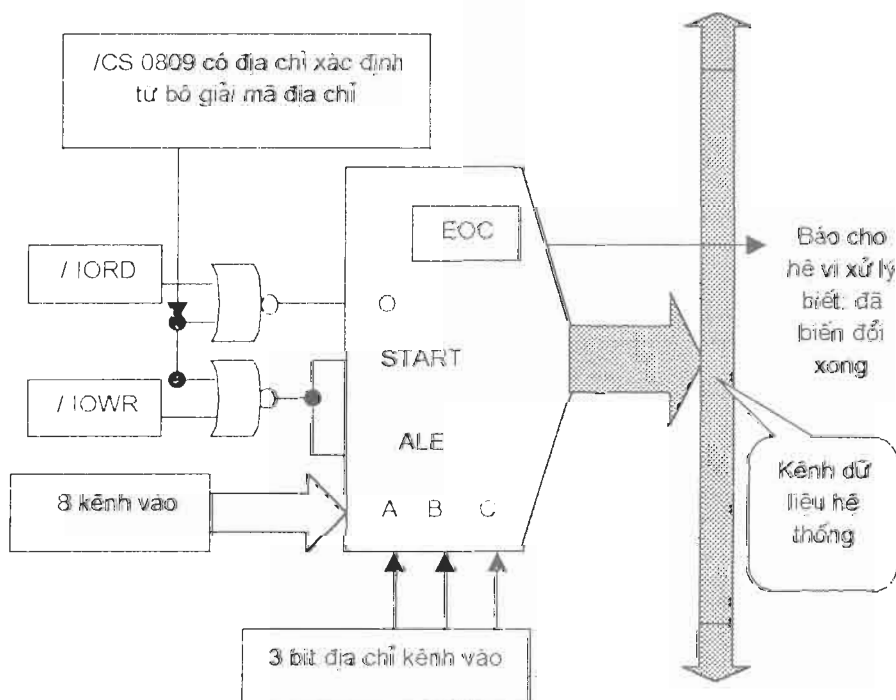
Hình 8.9. Đồ thị thời gian của ADC 0809

8.3.3. Ghép ADC 0809 với hệ vi xử lý

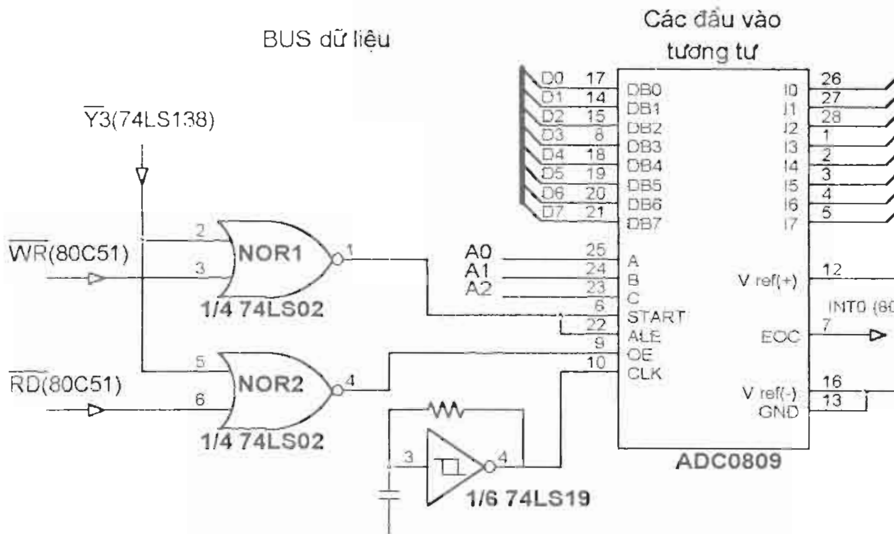
Ghép ADC 0809 với hệ vi xử lý được thể hiện trên hình 8.10. Các kênh vào analog được nối với các đầu vào tương ứng của ADC (có 8 kênh như vậy). Mỗi kênh có địa chỉ riêng do tổ hợp 3 bit địa chỉ ABC quy định. Các dây địa chỉ này có thể sử dụng trực tiếp các dây địa chỉ của kênh địa chỉ hệ vi xử lý. Thường là ABC của ADC 0809 được nối với A0A1A2 của hệ vi xử lý. Các dây địa chỉ cao của hệ vi xử lý được dùng để tạo tín hiệu chọn chip /CS cho ADC 0809. Tín hiệu /CS được đưa tới đầu vào của mạch OR để khởi động ADC (Start) khi có tín hiệu /IOWR đồng thời chốt địa chỉ (ALE) của kênh hiện hành có giá trị là giá trị ba bit ABC. Tín hiệu /CS cũng được đưa tới đầu vào của mạch OR thứ hai để tạo tín hiệu OE cùng với /IORD nhằm chốt dữ liệu đã biến đổi xong ở đầu ra.

Khi biến đổi xong, ADC 0809 dùng tín hiệu EOC để báo cho hệ vi xử lý biết mã nhị phân tương ứng với mức tín hiệu đầu vào đã được tạo ra. EOC thường được nối với đầu vào ngắt của CPU hay của PIC 8259. Trong một số trường hợp người ta dùng trễ thời gian để khẳng định quá trình biến đổi đã hoàn tất thì không cần sử dụng tín hiệu EOC.

8 bit dữ liệu thường được ghép trực tiếp với kênh dữ liệu hệ thống vì bản thân bộ đệm ra cũng là bộ đệm 3 trạng thái.

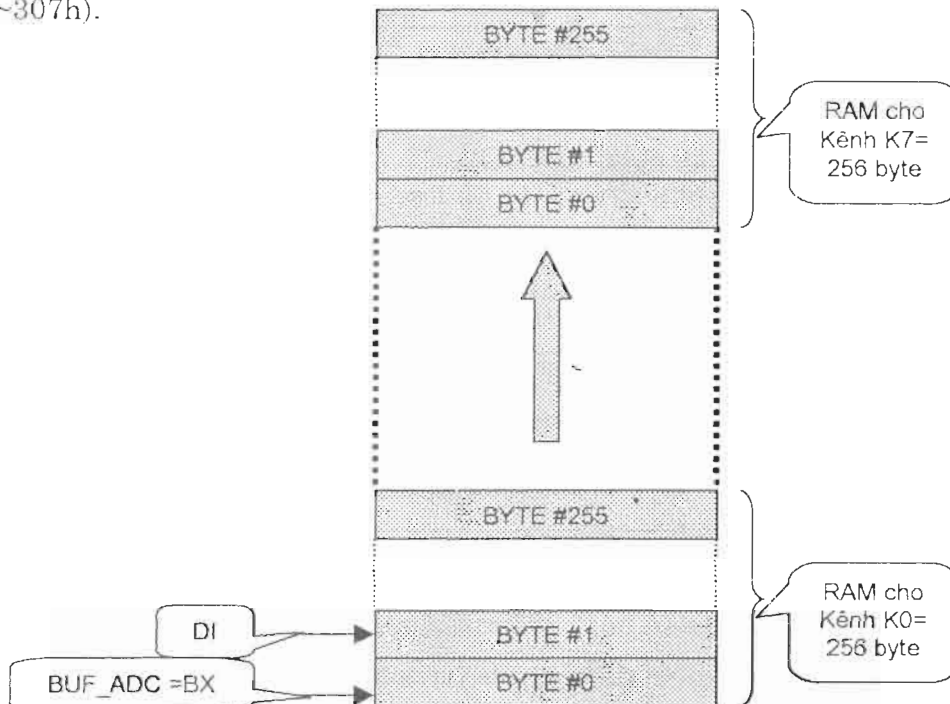


Hình 8.10 a. Ghép ADC 0809 với hệ vi xử lý



Hình 8.10 b. Ghép ADC 0809 với hệ vi xử lý

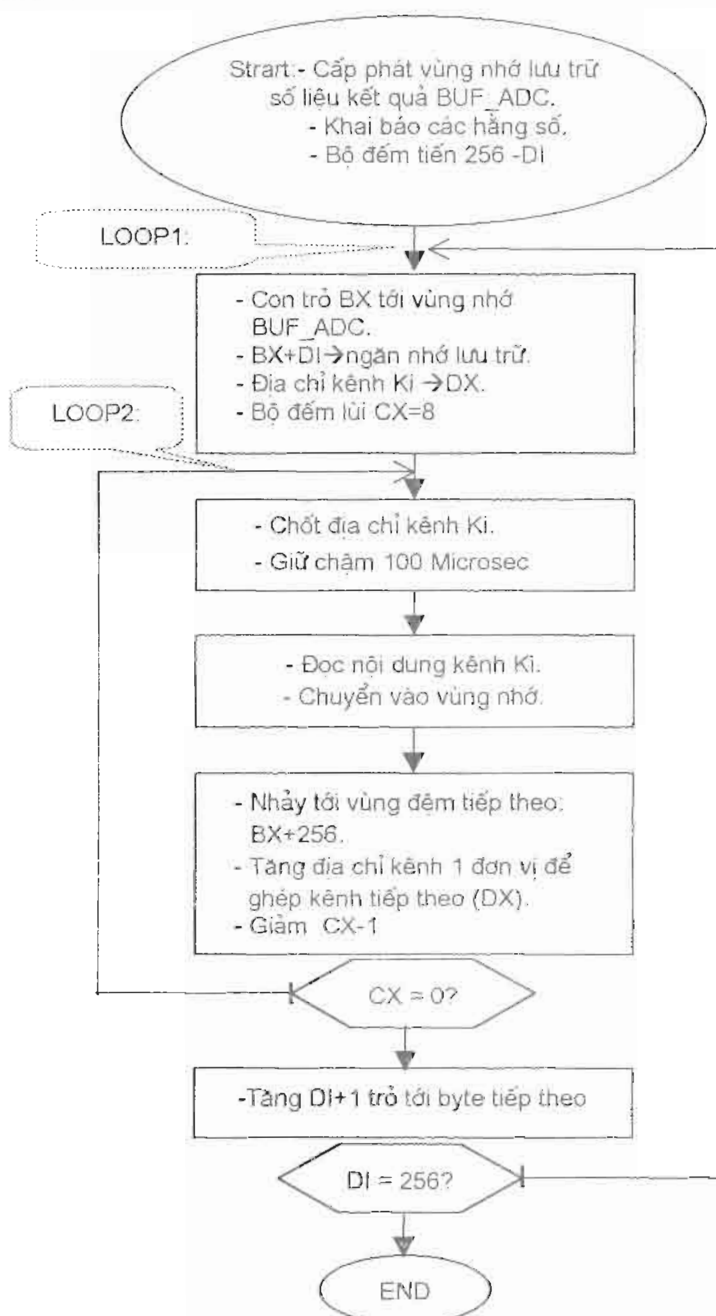
Chương trình minh hoạ cho sơ đồ ghép nối hình 8.10 a) và b) sẽ sử dụng phương pháp tạo trễ để khẳng định quá trình biến đổi ADC bằng thủ tục Delay_100_Microsec. Chương trình con Read_ADC khi được gọi sẽ điều khiển 8 kênh vào analog theo phương pháp quét vòng và thực hiện lấy 256 mẫu cho mỗi kênh. Dữ liệu này được lưu trữ trong vùng đệm BUF_ADC có dung lượng 8x256 byte. Cho rằng địa chỉ cơ sở của các kênh analog là 300h (ứng với kênh đầu tiên K0~300h còn K1~301h, K2~303h, K3~303h, K4~304h, K5~305h, K6~306h, K7~307h).



Hình 8.11. Vùng Ram BUF_ADC của 8 kênh ADC 0809

Như vậy vùng nhớ RAM BUF_ADC lưu trữ kết quả được biểu diễn theo sơ đồ 8.11. Vùng RAM cho mỗi kênh là 256 byte.

Thuật toán của chương trình được xây dựng trên 2 vòng lặp (hình 8.12). Vòng lặp ngoài sẽ thực hiện 256 lượt, mỗi lượt sẽ lưu trữ được 8 byte cho 8 kênh tương ứng. Bộ đếm tiến 256 sẽ kiểm soát chính xác số lượt lặp của chương trình. Vòng lặp trong sẽ thực hiện 8 lượt, mỗi lượt sẽ lưu trữ được 1 byte vào 1 kênh tương ứng. Bộ đếm lùi 8 sẽ kiểm soát chính xác số lượt lặp của modul chức năng. Bộ đếm lùi sẽ dùng cờ ZF để làm dấu hiệu thoát khỏi vòng lặp.



Hình 8.12. **Thuật toán chương trình điều khiển ADC 0809**

Chương trình nguồn Assembly có dạng như sau:

```
Cgroup GROUP CODE_SEG, DATA_SEG
ASSUME CS: Cgroup, DS: Cgroup;
        Chot    EQU 300H ; địa chỉ kênh đầu tiên
        SIZE_OF_BUFi EQU 256; kích thước bộ  đệm cho
                              ;từng kênh=256 byte
;--------------------------------------------------------------------
CODE_SEG SEGMENT
ORG 100H  ;để tạo file COM
;------------------------------------------
MAIN PROC
      CALL READ _ADC;
      ; các lệnh khác của thủ tục chính
MAIN ENDP
;------------------------------------------
READ_ADC PROC
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        PUSH DI
        MOV DI,00H ;bộ đếm tiến 256 -khởi đầu giá trị cho ngăn nhớ
                   ;vùng đệm
LOOP1:
        MOV BX,OFFSET BUF_ADC  ; địa chỉ bộ đệm kênh
        ADD BX,DI; BX trỏ tới ngăn nhớ này
        MOV DX,Chot ; lấy địa chỉ chốt của kênh đầu tiên
        MOV CX,08H  ; lặp cho 8 kênh
LOOP2:
        OUT DX, AL  ;khởi động kênh hiện hành
        CALL DELAY_100MicroSec  ; chờ ADC biến đổi xong
        IN AL,DX        ; đọc nội dung kênh hiện hành vào hệ VXL
        MOV [BX],AL; chuyển vào bộ đệm tương ứng
```

```
        ADD BX, SIZE_OF_BUFi; nhảy tới bộ đệm tiếp theo
        INC DX          ; tăng địa chỉ kênh 1 đơn vị
        LOOP LOOP2      ; lặp lại 8 kênh
        INC DI ; trỏ tới byte tiếp theo
        CMP DI,256
        JNZ LOOP1
        POP DI
        POP DX
        POP CX
        POP BX
        POP AX
        RET
READ_ADC ENDP
;---------------------------------------------
DELAY_100MicroSec PROC
; chương trình tạo trễ thời gian
DELAY_100MicroSec ENDP
;---------------------------------------------
CODE_SEG ENDS
;---------------------------------------------
DATA_SEG SEGMENT
        BUF_ADC  DB  8*256  DUP(?); dung lượng 8 bộ đệm kênh
DATA_SEG ENDS
;-----------------------------------------
END MAIN
```

8.4. BỘ BIẾN ĐỔI ADC 12 BIT AD574A

Để giảm sai số biến đổi tín hiệu tương tự-số đòi hỏi các bộ chuyển đổi ADC phải có độ phân giải cao. Ví dụ sử dụng bộ chuyển đổi ADC 8 bit trong mạch đo dòng điện ± 10A thì mỗi lượng tử tương ứng với mức $\frac{20}{2^8} \approx 78\,mA$ vậy sai số trong phạm vi 1 LSB tương ứng ± 0,19%. Nếu dùng bộ chuyển đổi ADC 12 bít thì sai số tương ứng sẽ giảm xuống còn ± 0,013%.

AD574A là ADC 12 bít. Vì AD574A chỉ có 1 đầu vào nên khi cần nhiều kênh analog phải sử dụng bộ dồn kênh tương tự. Đi theo với bộ dồn kênh tương tự và ADC là bộ giải mã địa chỉ để tạo địa chỉ phân biệt cho kênh tương tự đầu vào cũng như tạo ra các tín hiệu điều khiển trong quá trình chuyển đổi.

8.4.1. Cấu trúc của AD574A

Bộ chuyển đổi AD574A có các chỉ tiêu kỹ thuật sau:

- Độ phân giải 12 bít.
- Thời gian biến đổi : 25 μs.
- Trong dải nhiệt độ từ - 55°C đến +125°C chỉ có sai số tuyến tính,
- Tiêu thụ nguồn năng lượng thấp: 390 mW.

AD574A có một đầu vào tương tự và đầu ra 12 bit số, chuyển đổi theo phương pháp xấp xỉ liên tục, có bộ tạo chuẩn và bộ tạo xung nhịp được cấy bên trong. Đầu ra có bộ đệm ba trạng thái để ghép trực tiếp vào kênh dữ liệu của hệ vi xử lý.

AD574A được thiết kế với thành phần chính là 2 chip LSI gồm cả hai mạch tương tự và số, điều này tạo cho vi mạch có đặc tính linh hoạt và hiệu suất cao với tiêu hao năng lượng thấp nhất (hình 8.13):

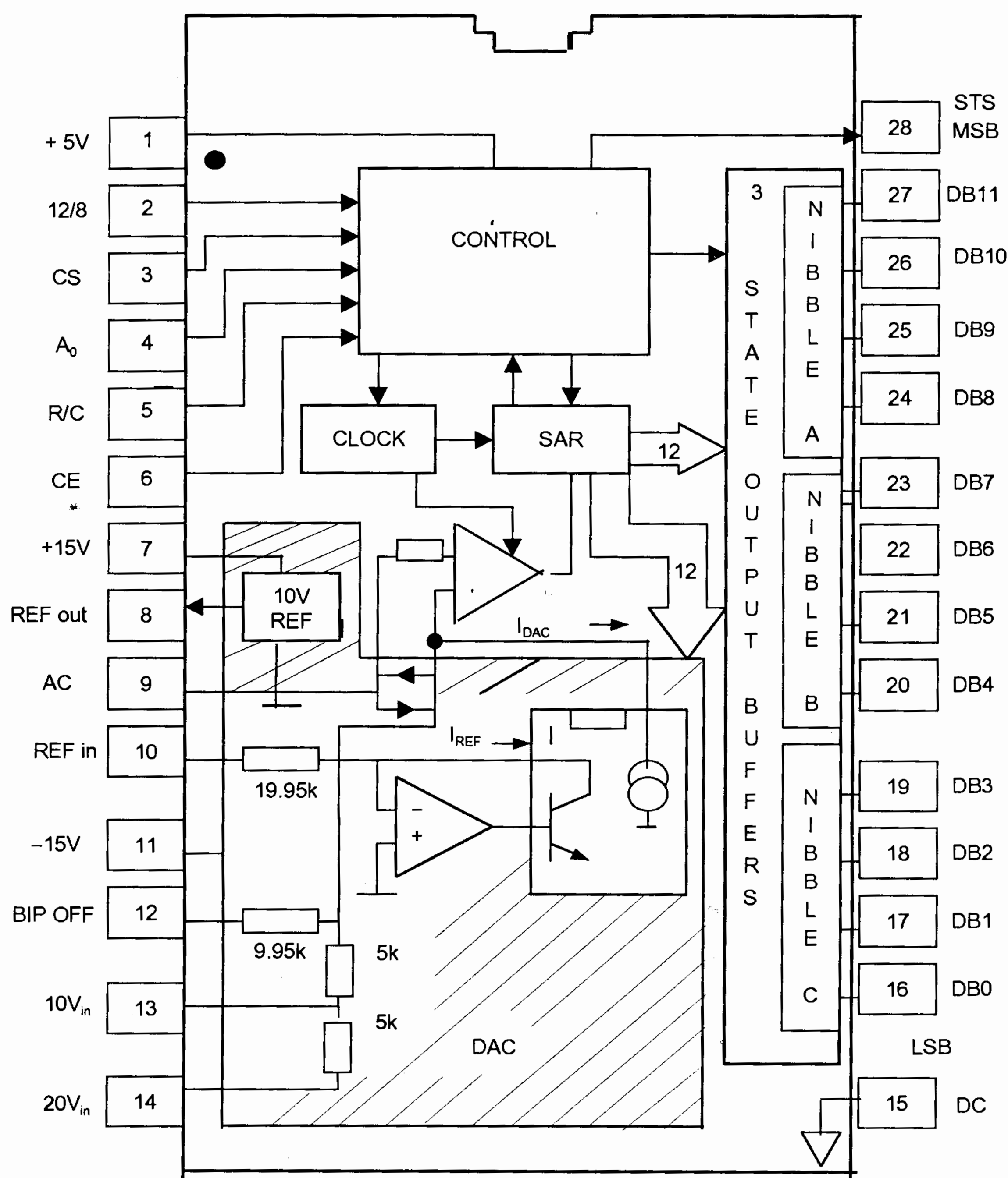
- Chip thứ nhất là bộ chuyển đổi ADC 12 bit AD565A hiệu suất cao và bộ tạo điện áp chuẩn. Nó còn có bộ chuyển mạch dòng đầu ra tốc độ cao và tập hợp điện trở màng mỏng được tinh chỉnh bằng Laser.
- Chip thứ hai sử dụng mạch LCI (Liner compatible integated injection logic) gồm thanh ghi xấp xỉ liên tục, bộ điều khiển lôgic, bộ đệm ba trạng thái, bộ tạo xung nhịp, và bộ hiệu chỉnh đầu vào, bộ so sánh có độ trôi thấp.

AD574A có thể ghép nối với một trong các kênh dữ liệu 8, 12 hoặc 16 bit của hệ vi xử lý không cần các bộ đệm cổng hoặc các phần tử điều khiển. Cả 12 bit dữ liệu đầu ra có thể đọc một lần hoặc hai lần mỗi lần 8 bit (lần đầu 8 bit dữ liệu, lần sau 4 bit dữ liệu kèm theo 4 bit zero).

Để đảm bảo tính chính xác khi tinh chỉnh, các điện trở bù lưỡng cực được chia thành bốn dải hiệu chỉnh: 0 đến +10V và 0 đến +20V đơn cực hoặc - 5V đến + 5V và -10V đến +10 V lưỡng cực, và hiệu chỉnh hết thang với sai số ± 0,1% có thể được tinh chỉnh đạt đến zero ứng với mỗi thành phần bên ngoài.

Diode ổn áp Zener bên trong được tinh chỉnh đến 10.000V với sai số cực đại 1%. Bộ hiệu chuẩn bổ sung bên ngoài có thể điều chỉnh dòng ở mức trên 1,5 mA nên yêu cầu có bộ chuẩn điện áp và các điện trở bù lưỡng cực.

Cấu trúc thành phần 2 chip làm cho AD574A trở nên tin cậy hơn trong các thiết kế lai ghép tạo nhiều kênh.



Hình 8.13. **Sơ đồ khối của AD574A**

Hoạt động của AD574A

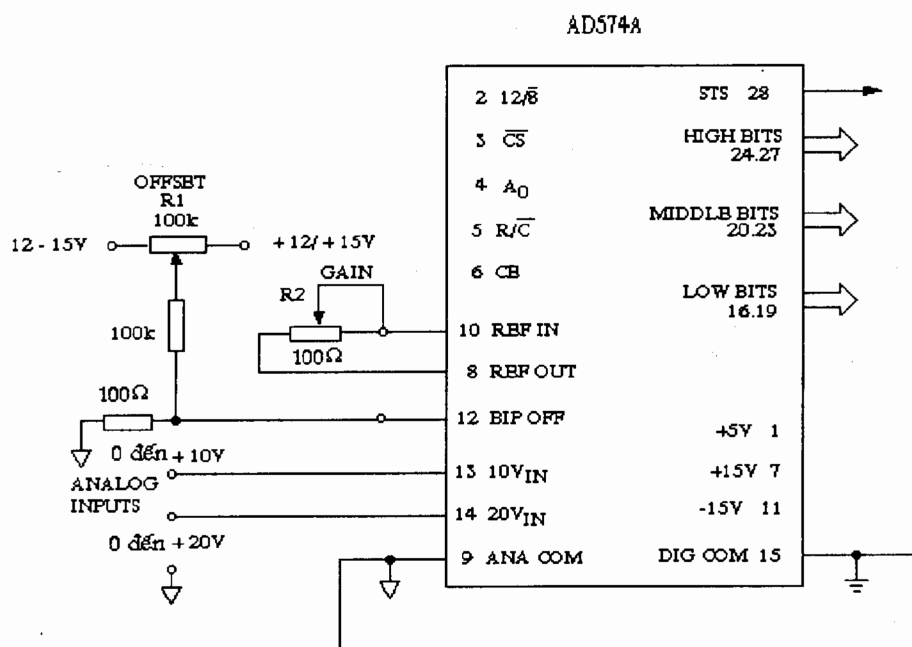
Khi AD574A nhận được lệnh điều khiển khởi đầu một chuyển đổi thì xung nhịp xoá nội dung thanh ghi xấp xỉ liên tục (SAR) về zero, đầu ra không có dữ liệu. Trong suốt chu trình chuyển đổi 12 bit đầu ra mạch DAC duy trì liên tục bởi SAR, từ bit có trọng số lớn nhất (MSB) đến bit có trọng số nhỏ nhất (LSB). Khối điều khiển sẽ dừng SAR khi đầu ra cờ trạng thái STS = 0

Bộ so sánh bù liên tục trọng số dòng cho mỗi bit tuỳ theo tổng dòng DAC lớn hoặc nhỏ hơn dòng đầu vào. Nếu dòng tổng nhỏ hơn thì bit bên trái được đặt lên 1, ngược lại thì bit đó được đặt là 0. Sau khi kiểm tra tất cả các bit SAR chứa 12 bít mã nhị phân ứng với tín hiệu đầu vào đã được xử lý xong thì cờ kết thúc sẽ bật lên.

Bộ so sánh chuẩn bù nhiệt zener cung cấp điện áp so sánh gốc cho bộ DAC và bảo đảm sự ổn định với cả hai biến thời gian và nhiệt độ. Bộ so sánh chuẩn ghim điện áp ở mức 10V ± 1%. Bất cứ chế độ tải nào của bộ chuẩn so sánh trong AD574A cần duy trì bằng hằng số trong suốt thời gian chuyển đổi. Sử dụng 2 điện trở điều chỉnh đầu vào 5 KΩ để cho phép sử dụng cả điện áp 10 V hoặc 20 V. Điện trở điều chỉnh 10 KΩ được nối đất để hoạt động ở chế độ đơn cực hoặc nối với điện áp 10 V để hoạt động ở chế độ lưỡng cực.

Hoạt động của AD574A khi tín hiệu đầu vào đơn cực

AD574A bao gồm tất cả các thành phần tích cực để thực hiện chuyển đổi 12 bit. Vì vậy trong hầu hết các trường hợp cần thiết đều phải nối với nguồn +5V, + 12V / +15V và -12V/ -15V. Các đầu vào tương tự được hiệu chỉnh dễ dàng. Sơ đồ làm việc khi đầu vào đơn cực như trên hình 8.14a. Ví dụ nếu không tinh chỉnh, AD574A đảm bảo sai số toàn thang ± 2LSB và bù trôi 0 cực đại là ± 3%. Nếu không tinh chỉnh bù trôi không thì nối trực tiếp chân 12 với chân 9, hoặc nếu không tinh chỉnh toàn thang thì dùng một điện trở 50Ω ± 1% nối giữa chân 8 và 10.



Hình 8.14.a

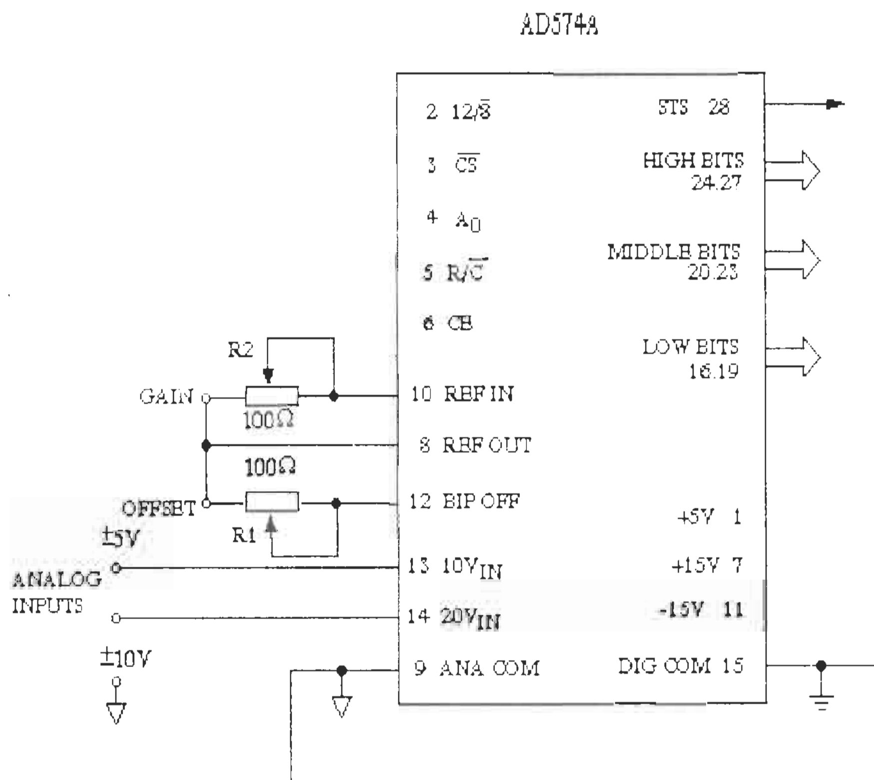
Đầu vào tương tự nối giữa chân 13 và 9, cho dải từ 0 đến +10V, giữa chân 14 và 9 cho dải từ 0 đến +20V. AD574A dễ phản ứng với tín hiệu vào vượt quá giá trị nguồn nuôi. Đầu vào thang 10 V giá trị mức LSB là 2,44 mV, thang 20V là 4,48 mV. Muốn có thang 10,24V định mức là 2,5 mV/ bit phải thay điện trở tinh chỉnh R_2 = 50 Ω. Dùng một điện trở 200 Ω nối tiếp chân 13 cho thang 20,48 V (5mV/bit). Điện trở tinh chỉnh 50 Ω nối tiếp chân 14 điều chỉnh hệ số khuếch đại. Trở kháng đầu vào định mức chân 13 là 5 kΩ, chân 14 là 10 kΩ.

Hiệu chuẩn đơn cực

AD574A được dự định bù định mức 1/2 LSB vì vậy đầu vào tương tự chính xác đối với mã nhận được sẽ ở chính giữa mã này. Khi hiệu chuẩn, đầu tiên cho đầu vào phù hợp mức ± 1/2 LSB (1.22 mV cho thang 10V và 2.44 mV cho thang 20 mV). Nếu chân 12 nối với chân 9 thì vẫn đảm bảo trong phạm vi các đặc tính cơ bản. Tinh chỉnh bù trôi 0 như trình bày ở trên. Một cách bù khác có thể thiết lập cho hệ thống có yêu cầu đặc biệt trong phạm vi tinh chỉnh xấp xỉ ± 15 mV. Tinh chỉnh toàn thang thực hiện bằng cách đặt một tín hiệu (1+1/2) LSB dưới định mức của toàn thang (9.9963 V cho thang 10V), tinh chỉnh R_2 để nhận được kết quả cuối cùng từ tổ hợp bit:1111 1111 1110 thành tổ hợp 1111 1111 1111.

Hoạt động của AD574A, khi tín hiệu đầu vào lưỡng cực

Sơ đồ nối các chân tín hiệu AD574A cho dải lưỡng cực được chỉ ra trên hình 8.14 b. Cũng như dải đơn cực, để đảm bảo các đặc tính khuếch đại và bù một cách đầy đủ thì có thể nối cứng một trong hai điện trở tinh chỉnh bằng 50Ω ± 1% . Đầu vào tương tự đặt như thang đơn cực. Việc định chuẩn được tiến hành cũng như định chuẩn đơn cực. Đầu tiên cung cấp tín hiệu 1/2 LSB trên cận thang (- 4.9988 cho thang ± 5V), rồi tinh chỉnh R_1 để nhận được chuyển đổi đầu tiên từ tổ hợp 0000 0000 0000 thành tổ hợp 0000 0000 0001. Sau đó một tín hiệu khác (1+1/2)LSB thấp hơn, cận dương + 4.9963 V (cho thang ± 5V) và tinh chỉnh R_2 để nhận được chuyển đổi cuối cùng từ tổ hợp 1111 1111 1110 thành tổ hợp 1111 1111 1111.



Hình 8.14.b

8.4.2. Điều khiển hoạt động AD574A

Khởi đầu chu trình chuyển đổi / Đọc dữ liệu AD574A bao gồm Logic on - chip để đảm bảo khởi tạo và đặt chế độ làm việc cho các thao tác đọc dữ liệu. Bảng sự thật của AD574A trong bảng 8.1

Bảng 8.1

CE	/CS	R-/C	12-/8	A_0	Chức năng
0	X	X	X	X	Không
X	1	X	X	X	Không
1	0	0	X	0	Chuyển đổi 12 bít
1	0	0	X	1	Chuyển đổi 8 bít đầu
1	0	1	Pin1	X	Cho phép đầu ra song song 12 bít
1	0	1	Pin15	0	Cho phép 8 MSB
1	0	1	Pin15	1	Cho phép 4 LSB kèm 4 bít 0

Các tín hiệu CE, /CS , và R -/C điều khiển hoạt động của bộ chuyển đổi. Trạng thái R-/C khi cả CE và /CS xác nhận bộ chuyển đổi đang đọc hay đang chuyển đổi. Khi đọc thì R-/ C = 1, còn khi chuyển đổi R-/C = 0. Các đầu vào thanh ghi A_0 và 12-/8 điều khiển dạng dữ liệu và độ dài chuyển đổi. Đường A_0

thường nối với bit LSB của Bus địa chỉ. Nếu việc chuyển đổi bắt đầu với $A_0 = 0$ thì ở đầu ra sẽ nhận được đủ cả 12 bít. Nếu $A_0 = 1$ thì sẽ chuyển đổi 8 bít thấp. Trong quá trình thao tác đọc dữ liệu, giá trị A_0 còn xác định bộ đệm ba trạng thái chứa 8 MSB (với $A_0 = 0$) hoặc 4 LSB (khi $A_0 = 1$).

Chân 12-/8 xác định dữ liệu đầu ra được sắp xếp thành 2 word 8 bit (12-/8 nối với chân DIG COM) hoặc một word đơn 12 bit (chân 12-/8 nối với V_{logic}). Chân 12-/8 không tương thích với TTL nên phải ghim về DIG hoặc V_{logic}.

Dạng thức 8 bít (khi $A_0 = 1$) chứa 4 LSB giá trị biến đổi kèm theo 4 bit zero. Cách tổ chức này cho phép các đường dữ liệu ra hoàn toàn trùng hợp và giao diện trực tiếp với Bus 8 bit mà không cần bộ đệm ba trạng thái.

Tín hiệu đầu ra STS chỉ thị trạng thái của bộ chuyển đổi. Khi STS = 1 là lúc bắt đầu chuyển đổi và STS = 0 khi chuyển đổi hoàn thành.

Đồ thị thời gian của AD574A

AD574A có giao diện trong phạm vi rộng với các hệ vi xử lý và các hệ thống số khác. Việc khảo sát các yêu cầu đặc tính thời gian, các tín hiệu điều khiển của nó sẽ giúp ích rất nhiều cho người thiết kế hệ thống. Bảng 8.2 mô tả các đặc tính thời gian cơ bản của AD574A.

Bảng 8.2. (Thời gian có đơn vị là ns)

Dạng chuyển đổi dữ liệu				Dạng đọc dữ liệu			
K.h	Tham số	Min	Max	K.h	Tham số	min	max
T_{DSC}	STS giữ chậm từ CE		300	t_{DD}	T.gian truy cập từ CE	210	250
T_{HEC}	Độ rộng xung CE	300		t_{HD}	Data hiệu lực khi CE=0	25	
T_{SSC}	/CS thiết lập từ CE	300		t_{HL}	Giữ chậm thả nổi đ. ra	110	150
t_{HSC}	/CS = 0 khi CE = 1	200		t_{SSR}	/CS thiết lập từ CE	150	
t_{SRC}	R/C thiết lập từ CE	250		t_{SRR}	R/ C thiết lập từ CE	0	
t_{HRC}	R/C = 0 khi CE = 1	200		t_{SAR}	A_0 thiết lập từ CE	150	
t_{SAC}	A_0 thiết lập từ CE	0		t_{HSR}	/CS hiệu lực sau CE=0	50	
t_{HAC}	A_0 hiệu lực khi CE=1	300		t_{HRR}	R/C =1 khi CE = 0	0	
t_c	Thời gian chuyển đổi			t_{HAR}	A_0 hiệu lực sau CE =0	50	
	Chu trình 8 bít	10^4	24.10^3				
	Chu trình 12 bít	15.10^3	35.10^3				

Dạng chuyển đổi dữ liệu

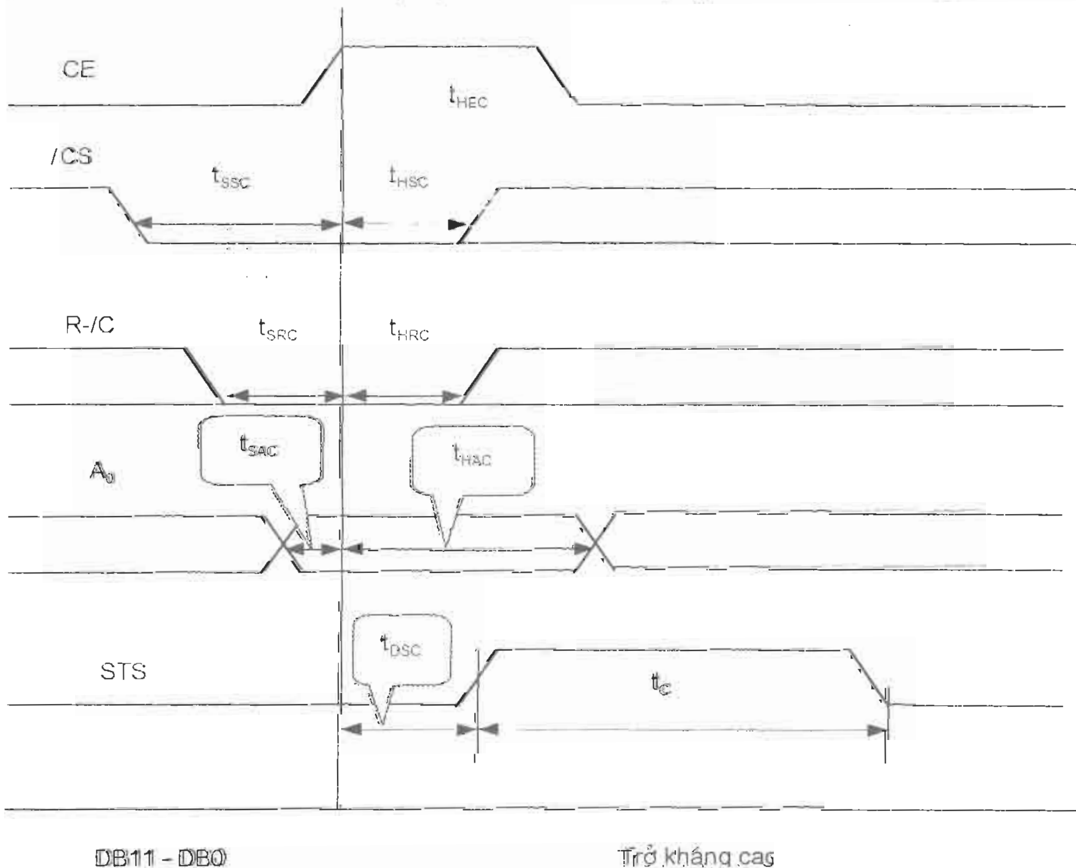
Đặc tính thời gian thao tác khởi đầu chuyển đổi của AD574A được thể hiện trên hình 8. 15. Chân R-/C cần ở mức thấp trước lúc CE và /CS xuất hiện. Nếu R-/C = 1 và thao tác đọc dữ liệu xuất hiện đồng thời có thể tạo thành sự tranh chấp trong Bus hệ thống. Có thể dùng CE hoặc /CS để khởi tạo một chuyển đổi. Trên hình 8.15 chỉ ra việc sử dụng CE cho chuyển đổi. Nếu sử dụng chân /CS để kích hoạt việc chuyển đổi A/ D hoặc không có thời gian thiết lập riêng, thì phải

kéo dài thời gian hiệu lực của các tín hiệu cần thiết (thời gian đảm bảo ít nhất là 200 ns trong khi R-/C, CE và /CS có hiệu lực). Đầu CE có thời gian trễ nhỏ hơn chân /CS nên nó là đầu vào nhanh hơn .

Khi bắt đầu chuyển đổi và đường STS được nâng lên mức High, các lệnh bắt đầu chuyển đổi mới sẽ bị bỏ qua, cho đến khi chu trình chuyển đổi được hoàn thành. Các bộ đệm dữ liệu đầu ra sẽ bị cấm trong quá trình chuyển đổi.

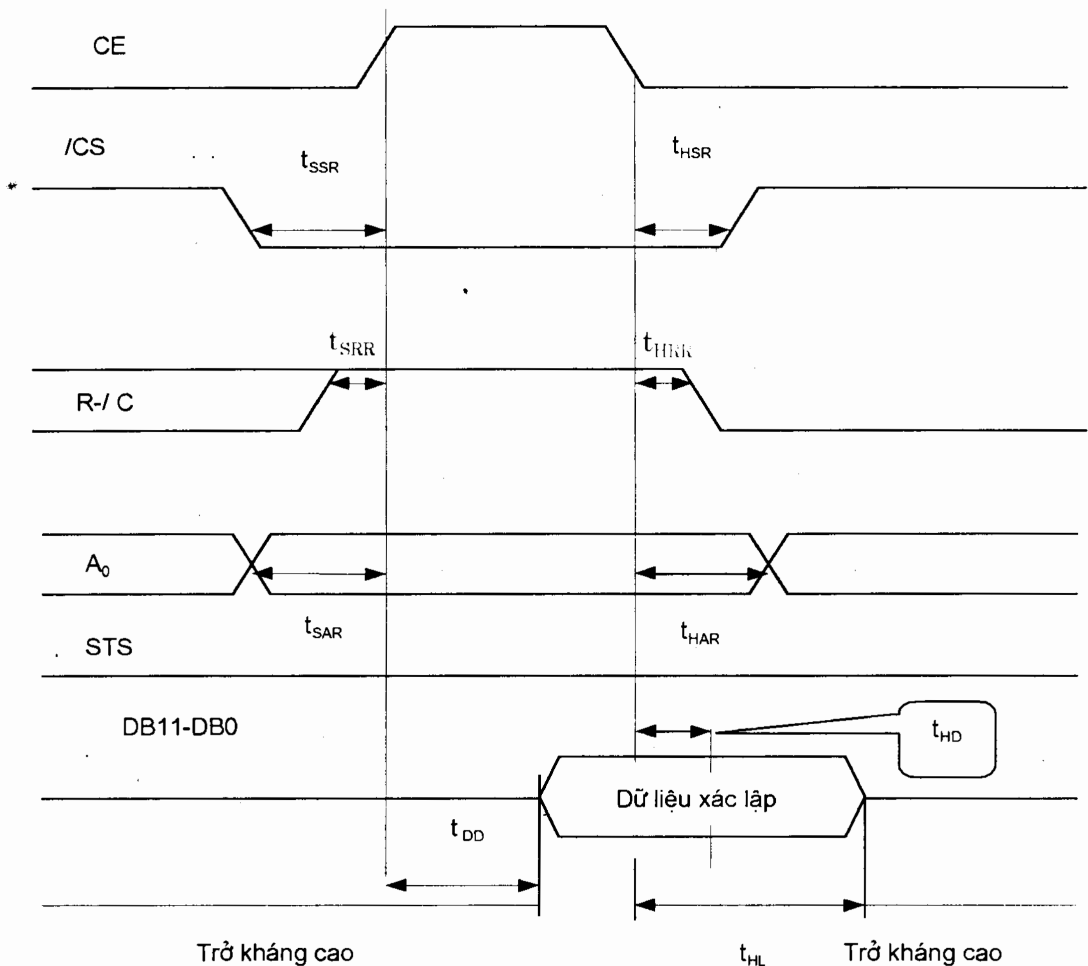
Dạng đọc dữ liệu

Đặc tính thời gian thao tác tiến hành đọc dữ liệu của AD574A được thể hiện trên hình 8.15. Bộ AD574A khác với bộ AD574 thế hệ cũ ở chỗ các bộ đệm đầu ra ba trạng thái có thời gian truy cập nhanh và thời gian ẩn dữ liệu ngắn hơn. Sự cải tiến tốc độ này để truy nhập nhanh với các hệ vi xử lý. Trong quá trình thao tác đọc dữ liệu, thời gian truy cập được tính từ thời điểm khi cả hai đường R-/C và CE ở trạng thái cao. Giả thiết /CS ở mức thấp mà dùng /CS để cho phép chọn chíp, thời gian truy cập kéo dài 100 ns. Đối với dạng giao diện thanh ghi 8 bít, bít địa chỉ A_0 phải ổn định ít nhất trong khoảng 150 ns, trước khi CE = 1 và giữ ổn định trong toàn bộ chu trình đọc. Nếu A_0 được phép thay đổi sẽ gây nguy hiểm cho bộ đệm đầu ra ba trạng thái của AD574A.



Hình 8.15. Đồ thị thời gian cho quá trình biến đổi AD

Trong thiết kế thực hiện nối chân 12-/ 8 với chân 15 khi cần tổ chức dữ liệu ra là dạng 8 bít. Việc ghép nối với Bus dữ liệu 8 bít được thực hiện trong định dạng căn lề trái. Địa chỉ chẵn (A_0 thấp) gồm 8 MSB (từ DB11 đến DB4). Địa chỉ lẻ (A_0 cao) gồm 4 LSB (Từ DB3 đến DB0) trong nửa đầu của byte, sau đó là 4 số 0. Ta không thể bố trí lại các đường dữ liệu của AD574A để ghép nối với Bus dữ liệu 8 bít dịch lề phải. Thời gian truy cập dữ liệu và thời gian chờ của các bộ đệm 3 trạng thái của vi mạch AD574A tương thích với các thiết bị nhớ sẵn có trên thị trường hiện nay. Vì thế vi mạch AD574A có thể ghép nối trực tiếp với Bus của bộ vi xử lý.



Hình 8.15. **Đồ thị thời gian cho quá trình đọc dữ liệu AD**

8.4.3. Ghép nối AD574A với hệ vi xử lý

Sơ đồ minh hoạ cho ghép nối AD574A với hệ vi xử lý được thể hiện trên hình C1 của phụ lục C. Căn cứ vào vùng địa chỉ mà hệ vi xử lý dành cho quản lý thiết bị ngoại vi là 16 bit địa chỉ A15-A0 ta dễ dàng tổ chức địa chỉ cho AD574A.

Theo qui định đó, trong sơ đồ tổ chức phần cứng ta giả thiết vùng địa chỉ được chọn là 300 ÷ 31F thì bộ giải mã có thể thiết kế theo phương án sau:

- Sử dụng một chip IC 74LS688 (comparator) làm chức năng so sánh. IC này có 16 đầu vào, chia đôi thành 2 vùng đầu vào P và Q, mỗi vùng có 8 bít thông tin. Đầu ra của 74LS688 chỉ có một, nó chỉ bằng 0 khi cả 8 bít ở hai vùng đầu vào P và Q giống nhau.
- Để xác định địa chỉ của từng chip ghép nối ta còn phải dùng bộ giải mã 74LS138. Để có tổ hợp 8 giá trị đầu vào của IC 74LS138 ta sử dụng 3 dây địa chỉ A2, A3, A4 sẽ tạo được 8 tổ hợp giá trị. Với một tổ hợp giá trị đầu vào thì chỉ có 1 trong 8 đầu ra của nó được tích cực mà thôi. Việc kết hợp giá trị đầu ra IC giải mã địa chỉ 74LS138 với giá trị đầu ra của IC 74LS688 giúp ta vừa tạo được vùng địa chỉ từ 300 ÷ 31F và đồng thời đã phân chia từng vùng địa chỉ riêng biệt cho các chip ghép nối.
- Các địa chỉ cần phải tổ chức (bảng 8.3):

+ Địa chỉ dồn kênh tương tự, đặt thang đo và bắt đầu quá trình đo: 300H

+ Một địa chỉ khởi động bộ AD574A : 304H

+ Một địa chỉ cho thao tác đọc dữ liệu ra của AD574A: 308H

+ Một địa chỉ chọn vi mạch 8255 : 30CH

Bảng 8.3

Các đường địa chỉ mã nhị phân từ hệ vi xử lý						Địa chỉ mã Hexa
A15-A10	A9 A8	A7 A6 A5	A4	A3 A2	A1 A0	
0 ... 0	1 1	0 0 0	0	0 0	0 0	300
0 ... 0	1 1	0 0 0	0	0 0	X X	300- 303
0 ... 0	1 1	0 0 0	0	0 1	X X	304 - 307
0 ... 0	1 1	0 0 0	0	1 0	X X	308 - 30B
0 ... 0	1 1	0 0 0	0	1 1	X X	30C- 30F
0 ... 0	1 1	0 0 0	1	0 0	X X	310 - 313
0 ... 0	1 1	0 0 0	1	0 1	X X	314 - 317
0 ... 0	1 1	0 0 0	1	1 0	X X	318 - 31B
0 ... 0	1 1	0 0 0	1	1 1	X X	31C- 31F
	74LS688					
Tổ hợp giải mã			địa chỉ 74LS138			

Tổ chức địa chỉ cho AD574A

Với các địa chỉ trên ta tổ chức các hoạt động chính của mạch ghép nối. Khi chuyển đổi dữ liệu các chân /CS và CE cho phép và đường địa chỉ Y_1 được đưa vào chân R-/C. Để phân biệt quá trình chuyển đổi và đọc dữ liệu, ta sử dụng hai cổng logic và một Triger D, xử lý hai địa chỉ Y_1 và Y_2 để nhận được các đường điều khiển R-/C, CE, /CS, A_0 (Hình 8.16).

Khi bắt đầu chuyển đổi dữ liệu, cổng địa chỉ Y_1 xuống thấp do đó

+ /CS xuống thấp.

+ CE lên cao.

+ Xoá Triger D làm cho Q xuống thấp đưa vào chân A_0($A_0 = 0$, khởi động dạng chuyển đổi 12 bít)

Việc đọc số liệu được tổ chức đọc hai lần theo địa chỉ cổng khi Y_2 xuốn thấp:

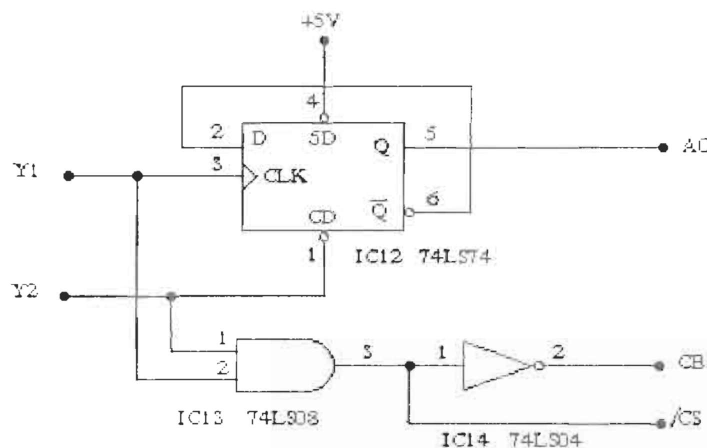
Lần 1 : /CS xuống thấp.

CE lên cao.

A_0 vẫn ở mức thấp.

Lần 2 : Sườn sau của Y_2 làm cho Triger chuyển trạng thái .

A_0 lên cao vì vậy ta sẽ tiếp tục đọc lần hai là 4 bít thấp của AD574A



Hình 8.16

Địa chỉ cho bộ chọn kênh đo

Từ bảng sự thật của bộ dồn kênh tương tự HEF 4051B và để đảm bảo nhận được chính xác dữ liệu từ các kênh analog ta sẽ tổ chức địa chỉ cho bộ chọn kênh đo như bảng 8.4.

Bảng 8.4

Các đầu vào				Các kênh analog							
/E	A2	A1	A0	Y0-Z	Y1-Z	Y2-Z	Y3-Z	Y4-Z	Y5-Z	Y6-Z	Y7-Z
L	L	L	L	ON	off	off	off	off	off	off	off
L	L	L	H	off	ON	off	off	off	off	off	off
L	L	H	L	off	off	ON	off	off	off	off	off
L	L	H	H	off	off	off	ON	off	off	off	off
L	H	L	L	off	off	off	off	ON	off	off	off
L	H	L	H	off	off	off	off	off	ON	off	off
L	H	H	L	off	off	off	off	off	off	ON	off
L	H	H	H	off	off	off	off	off	off	off	ON
H	X	X	X	off	off	off	off	off	off	off	off

Địa chỉ $300 đưa đến đầu vào cho phép /OE của bộ đệm chốt 74LS374. Tín hiệu /IOWR từ Bus của hệ vi xử lý được đưa đến đầu vào CLK của bộ đệm chốt dữ liệu từ D0 đến D7. Đầu ra ta sẽ sử dụng tổ hợp 3 tín hiệu từ Q0 đến Q2 để chọn 1 trong 8 kênh đo ở bộ dồn kênh tương tự HEF 4051B. Còn lại các đường từ Q3 đến Q7 có thể dùng để gán địa chỉ cho các chức năng khác. Trong mạch có sử dụng tổ hợp 2 tín hiệu từ Q3 đến Q4 để chọn thang đo. AD574A có thể tiếp nhận 8 kênh vào hoàn toàn độc lập, thông qua bộ chọn kênh IC1 và IC2, mỗi tín hiệu vào có hai dây vì vậy có thể độc lập với nhau. Trên mạch dùng hai bộ chọn kênh A và B, mỗi bộ có 8 đầu vào một đầu ra. Tuỳ thuộc vào tổ hợp của tín hiệu chọn kênh từ Q0 đến Q2 trong một thời điểm chỉ có một kênh được đưa đến bộ biến đổi AD.

Chọn thang đo

Điện áp đầu vào cho toàn dải biến đổi của IC7 AD574A là 0 ÷ ± 10V. Để có thể thay đổi dải đo nhằm tăng độ chính xác của phép đo trên mạch có dùng IC5 AD625 để khuếch đại, điện áp OUTA, OUTB được đưa từ bộ chọn kênh đo đến. Độ khuếch đại của IC5 được đặt bằng đầu ra Q3 và Q4 của IC3 74LS374 thông qua vi mạch Analog multiplexer IC6 Hi-509, tuỳ thuộc vào giá trị Q3 và Q4 đưa đến chân G0 và G1 của IC6 Hi-509 có thể đặt dải đo như sau:

Q3	Q4	Dải điện áp vào (V)
0	0	± 10.0
0	1	± 5.0
1	0	± 2.5
1	1	± 1.25

Chương 9
HỆ VI XỬ LÝ ON - CHIP

Nhờ sự phát triển của công nghệ điện tử, nhất là công nghệ chế tạo vi điện tử, đồng thời cùng với sự phát triển của các thế hệ máy vi tính, các hệ vi xử lý được tích hợp trong một chip (trong một vỏ IC- Microprocessor system on chip) bao gồm bộ vi xử lý Microprocessor, bộ nhớ chương trình EPROM, bộ nhớ dữ liệu RAM và bộ số học-logic ALU cùng với các thanh ghi chức năng, các cổng vào/ra, cơ chế điều khiển ngắt và truyền tin nối tiếp. Ngoài ra nó còn được trang bị các bộ thời gian Timer dùng trong các ứng dụng chia tần và tạo thời gian thực. Tương tự như hệ vi xử lý dùng bộ vi xử lý Intel-8085, Z80, CPU 80X86... bộ vi điều khiển có thể được lập trình để xử lý dạng đơn nhiệm ứng dụng điều khiển trong các thiết bị như thiết bị thông tin, viễn thông, thiết bị đo lường cũng như các ứng dụng trong công nghệ thông tin và kỹ thuật điều khiển tự động... hoặc ứng dụng điều khiển trong các thiết bị dân dụng khác.

Như vậy, On-chip 80C51 là bộ vi điều khiển, nó có đầy đủ chức năng của một hệ vi xử lý 8 bit, hoạt động ở tần số 12Mhz, với bộ nhớ EPROM (4Kb), RAM(128 byte) nội trú và có thể mở rộng bộ nhớ ra ngoài, có 4 cổng 8 bits vào ra 2 chiều để giao tiếp với thiết bị ngoại vi. Điểm đặc biệt của on-chip 80C51 là được điều khiển bởi một hệ lệnh của nó có số lệnh đủ mạnh, cho phép lập trình bằng ngôn ngữ Assembly là ngôn ngữ mạnh trong điều khiển hệ thống.

9.1. CẤU TRÚC CỦA HỆ VI XỬ LÝ ON-CHIP 80C51 (VÀ 89C51)

9.1.1. Cấu trúc chung của on-chip 80C51

Sơ đồ chân tín hiệu của on-chip được thể hiện như hình 9.1 với chức năng chân tín hiệu như sau:

RxD : Chân vào nhận tín hiệu nối tiếp.

TxD : Chân ra truyền tín hiệu nối tiếp.

/INT0 : Ngắt ngoài có số hiệu 0.

/INT1 : Ngắt ngoài có số hiệu 1.

T0 : Chân vào 0 của đồng hồ Timer 0.

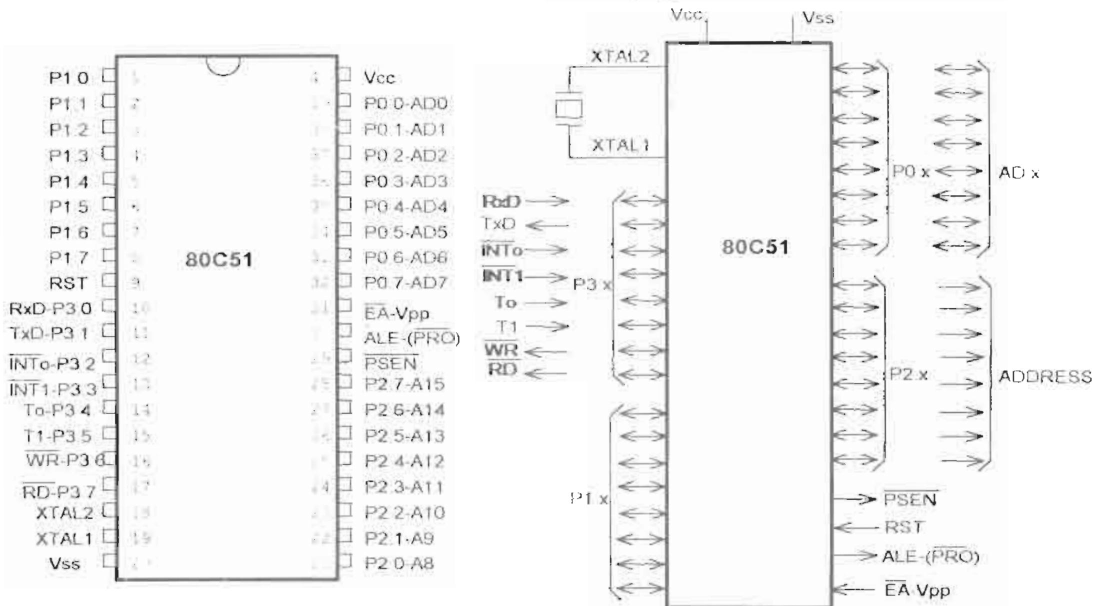
T1 : Chân vào 1 của đồng hồ Timer 1.

/WR : Ghi dữ liệu vào bộ nhớ ngoài.

/RD : Đọc dữ liệu từ bộ nhớ ngoài.

RST : Chân vào reset, tích cực ở mức cao trong khoảng 2 chu kỳ máy.

XTAL1 : Chân vào mạch khuếch đại dao động.



Hình 9.1. Sơ đồ chân tín hiệu của ON-CHIP 80C51

XTAL2 : Chân ra từ mạch khuếch đại dao động.

/PSEN : Chân cho phép đọc bộ nhớ chương trình ngoài (ROM ngoại trú). Khi on-chip làm việc với bộ nhớ chương trình ngoài, chân này phát ra tín hiệu kích hoạt ở mức thấp và được kích hoạt 2 lần trong mỗi chu kỳ máy. Chân /PSEN không được kích hoạt khi on-chip làm việc với bộ nhớ chương trình bên trong (ROM nội trú).

ALE(/PROG): Chân tín hiệu cho phép chốt địa chỉ khi on-chip xuất ra byte thấp của địa chỉ để truy cập bộ nhớ ngoài, tín hiệu chốt kích hoạt ở mức cao, tần số xung chốt <ALE> bằng1/6 fosc. Đây còn là chân nhận xung vào để nạp chương trình cho EPROM bên trong on-chip khi nó ở mức thấp .

/EA(Vpp): Chân cho phép lựa chọn làm việc với bộ nhớ chương trình, khi /EA = 1 cho phép on-chip làm việc với bộ nhớ chương trình bên trong và khi /EA = 0 thì cho phép làm việc với bộ nhớ chương trình bên ngoài. Khi chân này được cấp điện áp nguồn +21volt thì on-chip đảm nhiệm chức năng nạp chương trình cho EPROM bên trong nó.

Vcc : Chân cấp dương nguồn cho on-chip, dùng mức nguồn +5 volt .

Vss : Chân cấp âm nguồn, được nối mass (chân đất).

P0.x : Gồm các chân từ P0.0 đến P0.7 là chân của cổng 0

P1.x : Gồm các chân từ P1.0 đến P1.7 là chân của cổng 1

P2.x : Gồm các chân từ P2.0 đến P2.7 là chân của cổng 2

P3.x : Gồm các chân từ P3.0 đến P3.7 là chân của cổng 3. Các chân của cổng này ngoài nhiệm vụ xuất nhập dữ liệu qua cổng, còn đảm nhiệm chức năng điều khiển như hình 9.1.

9.1.2. Chức năng các thành phần của on-chip 80C51

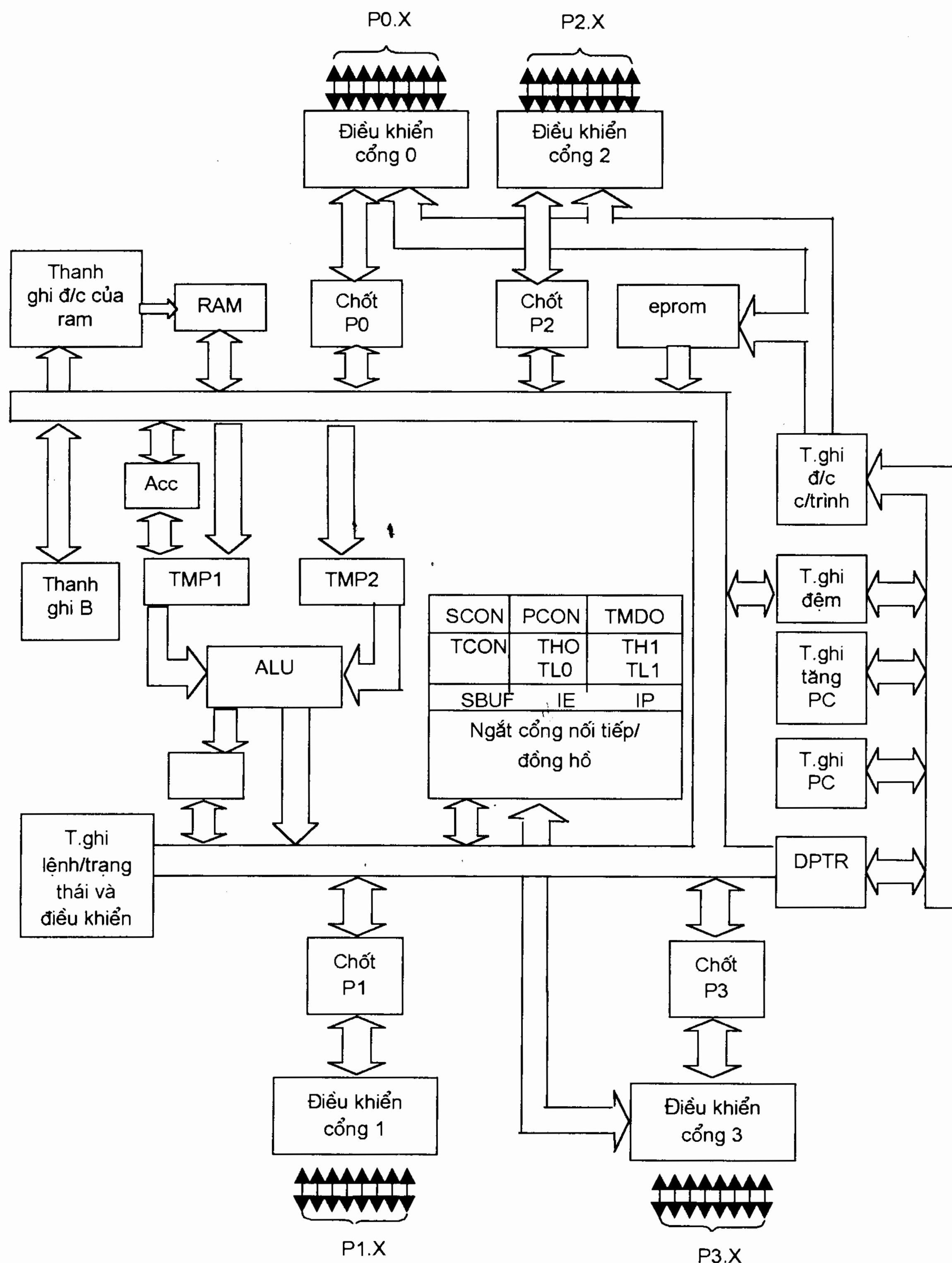
Cấu trúc phần cứng của on-chip được thể hiện trên hình 9.2. bao gồm :

- Các thanh ghi chức năng đặc biệt "special function registers" <SFR>.
- Bộ tính toán logic và số học <ALU>.
- Cổng vào ra <I/O>.
- Bộ nhớ chương trình và bộ nhớ dữ liệu <EPROM , RAM>.

Các thanh ghi chức năng đặc biệt

Các thanh ghi chức năng đặc biệt <SFR>là các thanh ghi đảm nhiệm các chức năng khác nhau trong on-chip. Chúng nằm ở RAM bên trong on-chip chiếm vùng không gian nhớ 128 byte được định địa chỉ từ 80h đến FFh. Cấu trúc của <SFR> trên hình 9.3 với chức năng:

- *Thanh ghi tích luỹ <Acc>(Accumulator)* : Đây là thanh ghi quan trọng trong on-chip, dùng để lưu trữ các toán hạng và kết quả của phép tính. Thanh ghi Acc dài 8 bits, có địa chỉ là E0h trong <SFR> .
- *Thanh ghi B* : Thanh ghi thường sử dụng khi thực hiện các phép toán nhân, chia. Đối với các lệnh khác, thanh ghi B có thể xem như là thanh ghi đệm tạm thời. Trong <SFR> thanh ghi B dài 8 bits có địa chỉ là F0h.
- *Con trỏ ngăn xếp <SP>(Stack Pointor)* : Thanh ghi này dài 8 bits, có địa chỉ trong <SFR> là 81h, giá trị của nó được tăng tự động trước khi thực hiện các lệnh PUSH, CALL. Ngăn xếp có thể đặt ở bất cứ nơi nào trong RAM on-chip, nhưng sau khi khởi động lại hệ thống thì con trỏ ngăn xếp mặc định sẽ trỏ tới địa chỉ khởi đầu là 07h, vậy ngăn xếp sẽ được tạo ra bắt đầu từ 08h.



Hình 9.2. **Sơ đồ chức năng của hệ VXL on-chip 80C51**

- *Con trỏ dữ liệu <DPTR>(Data Pointer)* : Là thanh ghi dài 16 bits gồm 2 thanh ghi dài 8 bits hợp lại là thanh ghi byte cao <DPH> và thanh ghi byte thấp <DPL>. Con trỏ dữ liệu có thể sử dụng như là thanh ghi 16 bit hoặc hai thanh ghi 8 bit độc lập. Trong <SFR> thanh ghi <DPH> có địa chỉ là 83h, còn thanh ghi <DPL> có địa chỉ là 82h .

F8									FF
F0	B								F7
E8									EF
E0	Acc								E7
D8									DF
D0	PSW								D7
C8									CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Hình 9.3. Tổ chức của thanh ghi chức năng đặc biệt <SFR> trong RAM.

- *Từ trạng thái chương trình <PSW> (Programmable status Word)*: Là thanh ghi dài 8 bits, có địa chỉ trong <SFR> là D0h, <PSW> dùng để chứa thông tin về trạng thái chương trình. Mỗi bit của <SFW> đảm nhiệm một chức năng cụ thể như hình 9.4 chỉ ra. Thanh ghi này được phép truy cập ở dạng mức bit.

- *Đệm dữ liệu truyền nối tiếp <SBUF>(Serial Buffer)* : Đệm dữ liệu truyền nối tiếp gồm 2 thanh ghi, một thanh ghi đệm phát và một thanh ghi đệm thu. Khi dữ liệu được truyền tới <SBUF>, nó được chuyển tới thanh ghi đệm phát và được giữ ở đấy để chế biến thành dạng truyền nối tiếp. Khi dữ liệu được truyền từ <SBUF> nó chuyển đi từ thanh ghi đệm thu. Địa chỉ của <SBUF> trong <SFR> là 99h.

♦ *Thanh ghi thời gian cơ sở* Timer: Trong on-chip 80C51 có hai đôi thanh ghi là TH0, TL0 và TH1, TL1. Mỗi đôi thanh ghi dài 16 bit dùng làm bộ đếm trong khối 'thời gian/ bộ đếm' tương ứng. Địa chỉ của các thanh ghi này trong <SFR> là từ 8Ah đến 8Dh.

msb lsb

CY	AC	FO	RS1	RS0	OV		P

Bit	kí hiệu	chức năng
7	CY	Cờ nhớ .
6	AC	Cờ nhớ phụ .
5	FO	Cờ 0 .
4	RS1	Bit 1 điều khiển chọn băng thanh ghi
3	RS0	Bit 0 điều khiển chọn băng thanh ghi
2	OV	Cờ tràn .
1	-	Bit dành cho người sử dụng định nghĩa .
0	P	Cờ chẵn lẻ .

* Trạng thái của 2 bit <RS1,RS0> dùng chọn băng thanh ghi như sau :

RS1	RS0	băng	địa chỉ
0	0	0	00h-:-07h
0	1	1	08h-:-0Fh
1	0	2	10h-:-17h
1	1	3	18h-:-1Fh

Hình 9.4. **Thanh ghi từ trạng thái chương trình PSW**

♦ *Thanh ghi điều khiển Mode 'thời gian / bộ đếm '<TMOD>(Timer Mode)*: Trong on-chip sử dụng kiểu 'thời gian/ bộ đếm', điều khiển kiểu chức năng 'thời gian' hay chức năng 'bộ đếm' bằng thanh ghi <TMOD>. Thanh ghi này dài 8 bit và có địa chỉ là 89h.

♦ *Thanh ghi điều khiển 'thời gian / bộ đếm '<TCON>(Timer Controller):* Tuy on-chip hoạt động theo kiểu 'thời gian' hay 'bộ đếm', nhưng trong mỗi kiểu lại có 4 chế độ hoạt động, việc chọn hoạt động ở 1 trong 4 chế độ được thực hiện bằng thanh ghi này. Trong <SFR> thanh ghi <TCON> có địa chỉ là 88h.

♦ *Thanh ghi điều khiển cổng nối tiếp <SCON>(Serial Controller)*: Thanh ghi này giúp on-chip thiết lập trạng thái và điều khiển cổng thực hiện chức năng truyền thông nối tiếp.Thanh ghi có địa chỉ là 98h trong <SFR>.

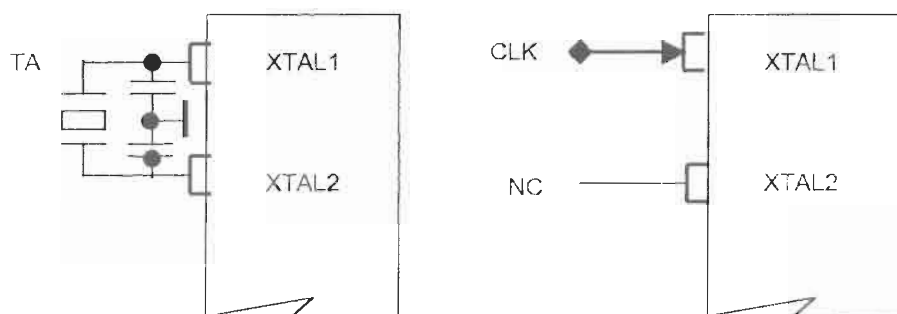
- *Thanh ghi cho phép ngắt <IE>(Interrupt Enable)*: Là thanh ghi dài một byte, có địa chỉ là A8h trong <SFR>. On-chip có khả năng truy cập địa chỉ mức bit tới thanh ghi <IE>. Trong thanh ghi này ngoài bit EA cho phép on-chip làm việc ở chế độ ngắt, còn các bit khác cho phép các ngắt tương ứng khác hoạt động. Khi bit có giá trị logic 0 thì ngắt tương ứng bị cấm, bit có mức logic 1 sẽ cho phép ngắt hoạt động .
- *Các cổng (PORT) 0,1,2 và 3 của on-chip*

  P0, P1, P2 và P3 là các thanh ghi đệm của các cổng 0, 1, 2 và 3 tương ứng, mỗi chốt <SFR> gồm 5 bit. Khi ghi mức logic 1 vào một bit của chốt thì chân ra tương ứng của cổng ở mức logic cao và ghi mức logic 0 vào mỗi bit của chốt thì chân ra tương ứng của cổng ở mức logic thấp. Khi các cổng đảm nhiệm chức năng như các đầu vào thì trạng thái bên ngoài của các chân cổng sẽ được giữ ở bit chốt <SFR> tương ứng. Tất cả 4 cổng của on-chip đều là cổng vào/ra 2 chiều, mỗi cổng gồm có 8 chân ra, mỗi chốt bit bên trong của nó có bộ 'pullup' do đó nâng cao khả năng nối ghép của cổng với tải (có thể giao tiếp với 4 đến 8 tải loại TTL). Cấu trúc và chức năng hoạt động của các cổng được trình bày ở phần sau .

Bộ tạo dao động của on-chip

On-chip có 2 chân <XTAL1>và <XTAL2> được dùng nối với bộ dao động để tạo xung đồng hồ cho on-chip. Có 2 phương pháp tạo xung đồng hồ cho on-chip đó là dùng dao động trong và dùng tín hiệu dao động ngoài.

On-chip sử dụng bộ dao động trong bằng cách nối 2 chân <XTAL1>và <XTAL2> với một mạch cộng hưởng thạch anh TA có tụ thoát nhiễu xuống đất, hoặc sử dụng tín hiệu dao động ngoài đưa vào on-chip qua chân<XTAL1> như hình thể hiện trên 9.5.

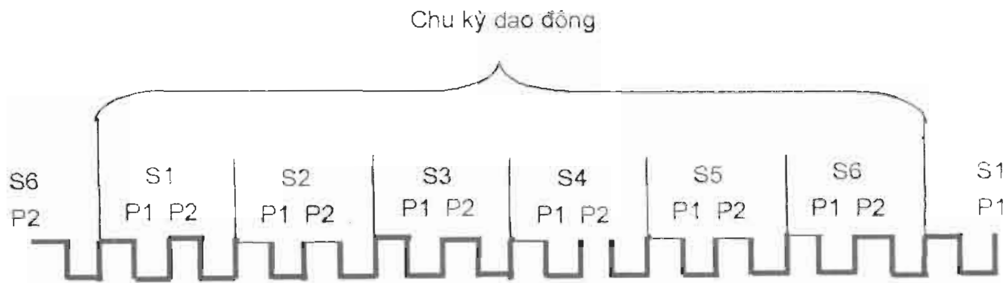


Hình 9.5. Tạo xung đồng hồ cho on-chip

Chu kỳ máy của On-chip

Một chu kỳ máy của on-chip 80C51 bao gồm 6 trạng thái từ S1 đến S6, mỗi trạng thái gồm 2 nhịp và được gọi là pha P1, P2. Như vậy mỗi chu kỳ máy

có 12 nhịp (pha). Nếu bộ dao động làm việc ở tần số 12Mhz thì 1 pha kéo dài 1μs . Minh hoạ chu kỳ máy trên hình 9.6.



Hình ***9.6.*** Minh hoạ chu kỳ máy trên

9.2. TỔ CHỨC CỔNG VÀO/RA CỦA HỆ VI XỬ LÝ ON-CHIP

Cấu hình vào ra <I/O>

Mỗi cổng của on-chip 80C51 chứa 1 chốt <SFR> một bộ điều khiển ra và một bộ đệm vào. Bộ điều khiển ra của cổng 0 và cổng 2 và bộ đệm vào của cổng 0 được dùng để truy cập bộ nhớ ngoài, cổng 0 xuất ra byte thấp của địa chỉ, cổng 2 xuất ra byte cao của địa chỉ khi cần địa chỉ hoá cho bộ nhớ ngoài. Ngoài chức năng truy cập địa chỉ, cổng 0 còn truyền dữ liệu 2 chiều vào/ra. Cổng 3 là cổng đa chức năng, chúng không những là cổng xuất nhập dữ liệu mà còn làm chức năng điều khiển đặc biệt khác như :

Chân P3.0 : <RxD>(Receiver Data Port) là cổng vào nối tiếp .

Chân P3.1 : <TxD>(Transmitter Data Port) là cổng ra nối tiếp .

Chân P3.2 : <INT0>(Interrupt0) là chân ngắt ngoài 0 .

Chân P3.3 : <INT1> (Interrupt1) là chân ngắt ngoài 1 .

Chân P3.4 : <T0> (Timer0) là chân vào cho Timer 0 .

Chân P3.5 : <T1> (Timer1) là chân vào cho Timer 1 .

Chân P3.6 : </WR>(Write command) là chân điều khiển ghi dữ liệu vào bộ nhớ ngoài.

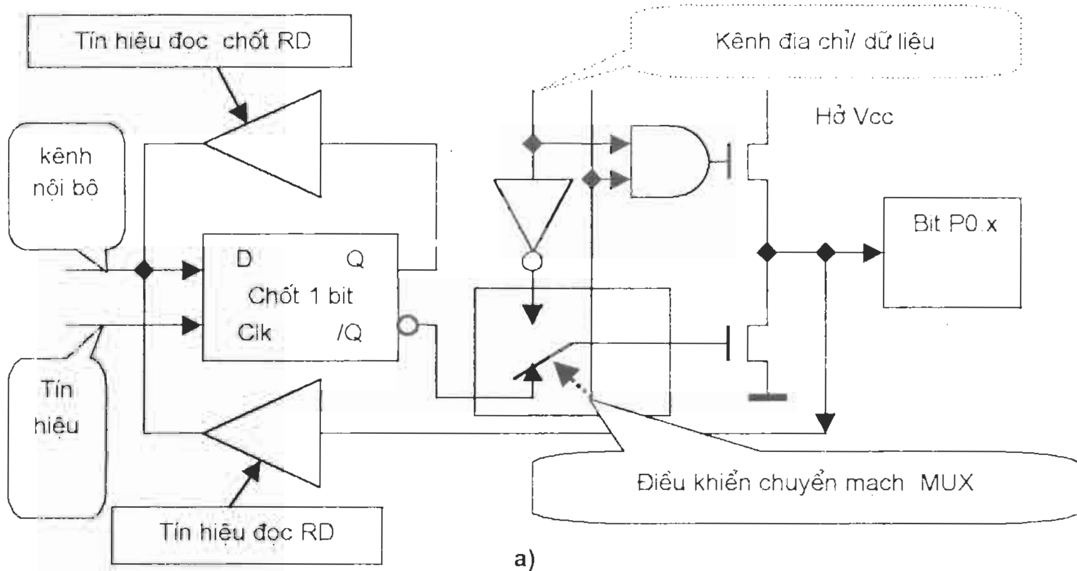
Chân P3.7 : </RD>(Read command) là chân điều khiển đọc dữ liệu từ bộ nhớ ngoài.

Các chức năng trên được phép hoạt động khi chốt bit tương ứng <SFR> ở mức logic 1.

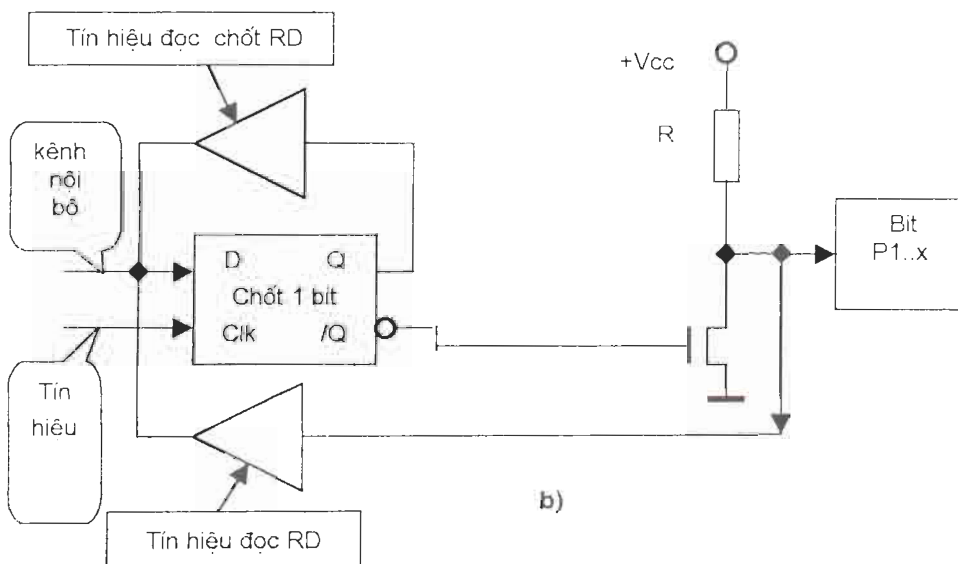
Hình 9.7 biểu diễn sơ đồ chức năng của một bộ đệm vào/ra và một chốt bit tiêu biểu của cả 4 cổng. Chốt bit <SFR> của cổng được miêu tả như là một flip-flop loại D, nó tạo xung nhịp CLK để ghi một giá trị từ kênh nội bộ để đáp ứng

cho tín hiệu ghi WR từ CPU. Mức logic của chân tín hiệu cổng sẽ được đưa vào kênh nội bộ để trả lời cho tín hiệu đọc RD từ CPU . Một số lệnh đọc cổng sẽ kích hoạt tín hiệu đọc chốt (đọc đầu ra của Trigơ D của bit chốt), và các lệnh khác thì kích hoạt tín hiệu đọc chân. Điều khiển ra của cổng 0 và cổng 2 được thực hiện bằng tín hiệu điều khiển chuyển mạch bên trong, để điều khiển giữa kênh địa chỉ và địa chỉ / dữ liệu khi truy cập bộ nhớ ngoài.

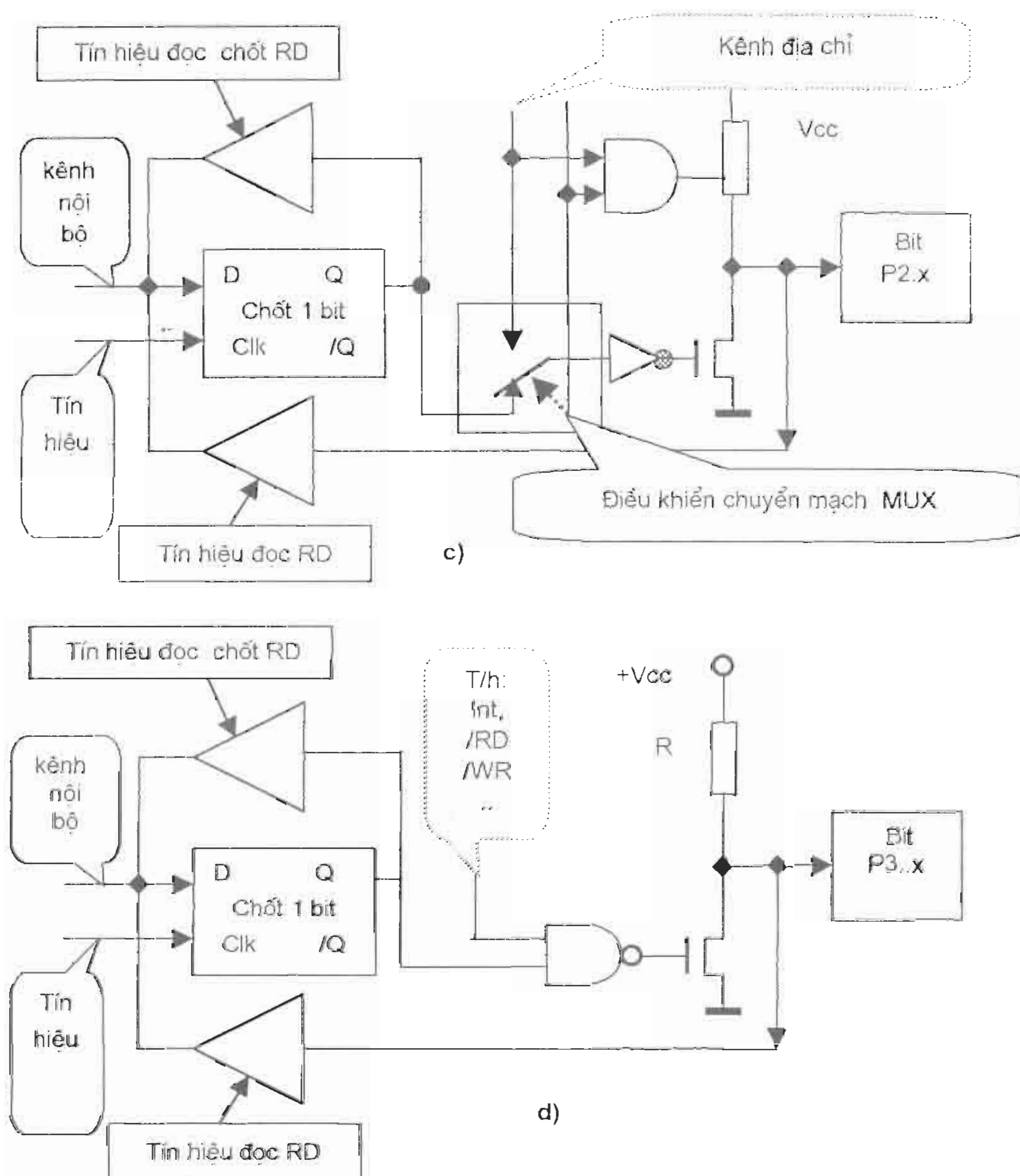
Nếu chốt bit P3 chứa giá trị 1 thì mức logic ra được điều khiển bởi tín hiệu "chức năng ra luân phiên " có nghĩa là các chân tín hiệu của P3 vừa có thể làm chức năng cổng vào/ra thông thường vừa có khả năng thực hiện chức năng khác như làm cổng truyền tin nối tiếp hay làm đầu vào ngắt...



a)



b)



Hình 9.7. Cấu trúc cổng a) P0, b) P1, c) P2, d) P3

Các cổng 1, 2 và 3 có các bộ khuyếch đại công suất bên trong. Đối với cổng 0 các chốt bit có bộ khuyếch đại công suất đẩy-kéo bên trong và là các mạch hở collector nên cần có điện trở ghép nguồn dòng bên ngoài.

Thao tác ghi thông tin vào cổng

Khi thực hiện một lệnh mà làm thay đổi giá trị của chốt cổng thì giá trị mới này được chốt vào cổng tại thời gian P2-S6 trong chu kỳ cuối của chu trình lệnh. Tuy nhiên giá trị trích mẫu ở các chốt cổng là giá trị của các bộ đệm ra

xuất hiện chỉ trong trong các pha 1 (P1) của chu kỳ máy, sau đó chuyển đến chốt cổng vào pha thứ 2 của trạng thái thứ 6 (P2-S6), bởi vậy giá trị mới không xuất hiện ở chân ra ngay trong và sau pha 1. Nếu sự thay đổi yêu cầu chuyển tiếp từ 0 sang 1 trong cổng 1, 2 và 3 thì sẽ có sự tham gia của 1 bộ khuyếch đại dòng (pullup) trong thời gian P1-S1 và P2-S1 của chu kỳ máy ở cổng có sự chuyển tiếp. Bộ khuyếch đại dòng (pullup) là các transistor trường, do đó nó có thể tạo ra dòng gấp 100 lần bộ khuyếch đại dòng bình thường.

9.3. KHỐI TẠO THỜI GIAN VÀ BỘ ĐẾM (TIMER/COUNTER)

Thanh ghi thời gian và bộ đếm

Trong on-chip có 2 thanh ghi dài 16 bit có thể hoạt động như kiểu bộ tạo thời gian hay hoạt động như kiểu bộ đếm. Thanh ghi TMOD trong <SFR> sẽ điều khiển kiểu hoạt động này. Cấu trúc và trạng thái của thanh ghi TMOD được thể hiện như hình 9.8.

Khi thanh ghi 'thời gian/ bộ đếm' làm việc ở kiểu bộ thời gian thì sau mỗi chu kỳ máy nội dung trong thanh ghi được gia tăng thêm 1 đơn vị. Một chu kỳ máy có 12 chu kỳ dao động, do đó tốc độ tăng của thanh ghi là 1/12 tần số dao động.

MSB LSB

GATE	C/T	M1	M0	GATE	C/T	M1	M0

GATE: là bit tham gia điều khiển cơ chế chuyển mạch cho Timer/Counter (xem hình 9.10, 9.11 và 9.12).

C/T : là bit thực hiện chuyển mạch cho Timer (đồng hồ) làm việc nếu được gán bằng 0 (/T) hoặc cho Counter (bộ đếm) làm việc nếu được gán bằng 1 (T).

M1 M0 : Là 2 bit xác định trạng thái của bộ tạo thời gian/ bộ đếm như sau:

M1	M0	Tác dụng
0	0	Dùng như với bộ thời gian 8048 TLx dài 5 bits
0	1	Các thanh ghi TLx, THx dài 16 bits được ghép tầng.
1	0	Các thanh ghi tự nạp lại mỗi khi bị tràn , trong đó thanh ghi THx dài 8 bits chứa giá trị nạp lại cho TLx
1	1	Thanh ghi TL0 được điều khiển bằng các bits của thanh ghi TL1

Hình 9.8. Cấu trúc của thanh ghi TMOD

<H-1.8> Thanh ghi điều khiển kiểu "thời gian / bộ đếm"

**Nguyên lý hoạt động của "thời gian/ bộ đếm"*

MSB							LSB
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Bit	Kí hiệu	Chức năng
7	TF1	(Timer Flag): Cờ tràn cho bộ tạo thời gian / bộ đếm 1 được thiết lập mức logic 1 bằng phần cứng. Nó được xoá bằng phần cứng khi bộ vi xử lý gọi chương trình con phục vụ ngắt, hoặc được xoá bằng phần mềm.
6	TR1	(Timer run): Bit điều khiển thời gian / bộ đếm 1 hoạt động, được đặt xoá bằng phần mềm .
5	TF0	Cờ tràn cho bộ tạo thời gian/ bộ đếm 0, được thiết lập mức logic1 bằng phần cứng . Nó được xoá bằng phần cứng khi bộ vi xử lý gọi chương trình con phục vụ ngắt, hoặc được xoá bằng phần mềm.
4	TR0	Bit điều khiển thời gian / bộ đếm 0 hoạt động. Nó được đặt xoá bằng phần mềm
3	IE1	(Interrupt external 1) Cờ ngắt 1 được thiết lập bằng phần cứng và tác động tích cực bằng sườn dương của tín hiệu ngắt INT1 . Cờ ngắt được xoá khi ngắt được xử lý .
2	IT1	Bit điều khiển đầu ngắt 1 cách kích hoạt bằng sườn (IT1 = 1) hoặc bằng mức (IT1 = 0). Nó được đặt/ xoá bằng phần mềm .
1	IE0	Cờ ngắt 0 được thiết lập bằng phần cứng và tác động tích cực bằng sườn dương của tín hiệu ngắt INT0. Cờ ngắt được xoá khi ngắt được xử lí .
0	IT0	Bit điều khiển đầu ngắt 0 cách kích hoạt bằng sườn (IT0 = 1) hoặc bằng mức (IT0 = 0). Nó được đặt/ xoá bằng phần mềm.

Hình 9.9. Thanh ghi <TCON> của <SFR>

Thanh ghi "thời gian /bộ đếm" khi làm việc ở kiểu bộ đếm thì tại thời điểm P2-S5 của mỗi chu kỳ máy, xung nhịp bên ngoài được đưa vào để đếm theo chân T0 hoặc T1 khi chuyển từ mức 1 xuống mức 0 sẽ làm cho thanh ghi gia tăng ở thời điểm P1-S3 của mỗi chu kỳ máy tiếp theo. Vậy nội dung của thanh ghi tăng thêm 1 đơn vị phải mất 2 chu kỳ máy, cho nên tốc độ đếm của thanh ghi là 1/24 tần số dao động.

Do xung nhịp bên ngoài có tần số bất kỳ, vì vậy "thời gian / bộ đếm" của on-chip có 4 chế độ hoạt động để chọn lựa .

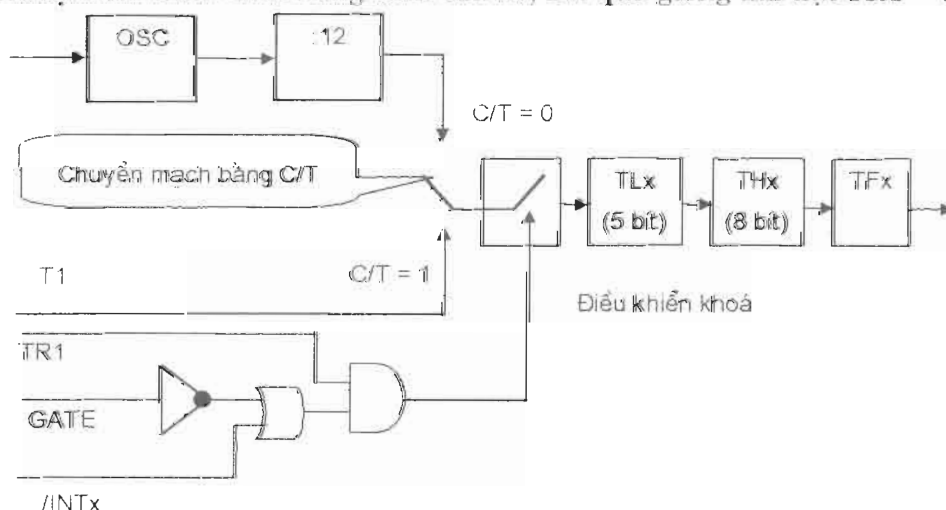
Chức năng "thời gian" hoặc "bộ đếm" được chọn lựa bởi các bit điều khiển C/T trong TMOD. Mỗi chức năng có 4 chế độ hoạt động lại được lựa chọn bởi cặp bit M1M0. Các chế độ hoạt động 0, 1 và 2 giống nhau cho cả chức năng "thời gian / bộ đếm 1", riêng chế độ 3 thì khác nhau "Thời gian / bộ đếm" lại được điều khiển bằng các bit trong thanh ghi <TCON> của <SFR> như thể hiện trên hình 9.9.

+ Chế độ 0 được mô tả trên hình 9.10 biểu diễn hoạt động của "thời gian /bộ đếm" 0 và 1 ở chế độ này Trong chế độ 0, thanh ghi "thời gian / bộ đếm" có

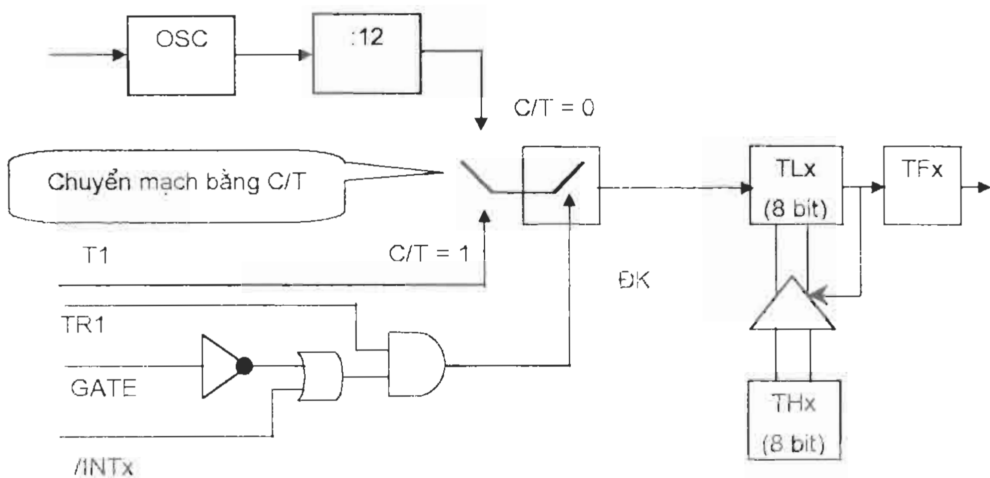
cấu hình như một thanh ghi 13 bit, trong đó thanh ghi THx sử dụng cả 8 bit còn thanh ghi TLx chỉ sử dụng 5 bit thấp. Khi thanh ghi được xoá về 0, cờ ngắt TFx được thiết lập. Khi chức năng của "Thời gian / bộ đếm" hoạt động thì bit điều khiển TRx được thiết lập ở mức logic 1, bit GATE trong <TMOD> bằng 0 và bit điều khiển ngắt ngoài /INTx bằng 1, nếu GATE = 1 thì cho phép điều khiển "thời gian /bộ đếm" bằng đường vào ngoài /INTx = 0

Khi hoạt động ở chức năng "thời gian" thì bit C/T = 0 do vậy xung nhịp từ bộ dao động nội, qua bộ chia tần cho ra tần số f = fosc/12 được đưa vào để đếm trong thanh ghi "thời gian / bộ đếm". Khi hoạt động ở chức năng "bộ đếm" thì bit C/T = 1 khi đó xung nhịp ngoài đưa vào sẽ được đếm.

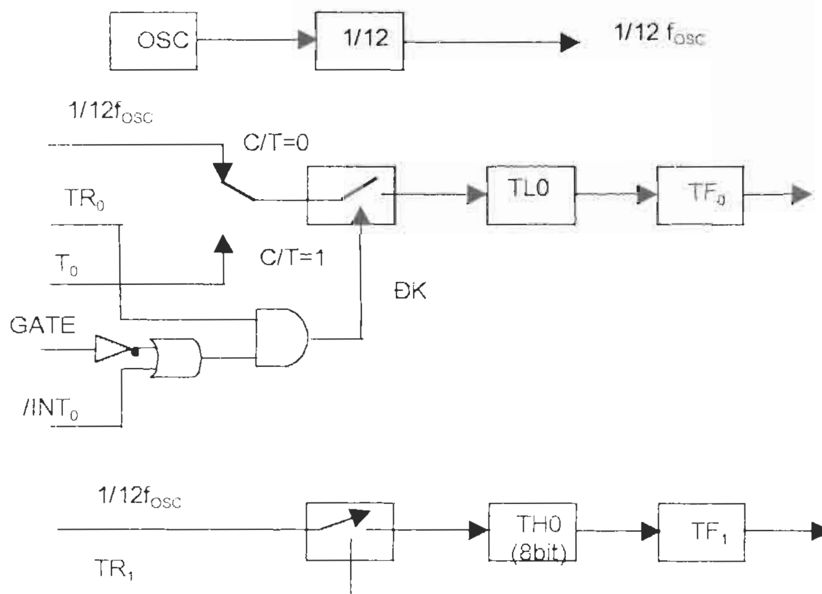
+ Chế độ 1 hoạt động tương tự như chế độ 0, chỉ khác là thanh ghi "thời gian / bộ đếm" được sử dụng cả 16 bit .
+ Chế độ 2 hoạt động tương tự như hai chế độ trên nhưng cấu trúc của thanh ghi "thời gian/bộ đếm" khác với 2 chế độ trên. Thanh ghi "thời gian/ bộ đếm" chỉ sử dụng 8 bit của TLx, khi TLx đếm tràn không chỉ làm cờ TFx được thiết lập mà còn tự động nạp lại cho TLx bằng nội dung của thanh ghi THx đã được thiết lập bằng phần mềm như thể hiện trên hình 9.11.
+ Chế độ 3: ở chế độ này chức năng "thời gian / bộ đếm 0" và chức năng "thời gian / bộ đếm 1" khác nhau, sơ đồ cấu trúc như thể hiện trên hình 9.12. "Thời gian / bộ đếm 0" ở chế độ 3 thiết lập TL0, TH0 như là 2 bộ đếm riêng biệt. Bộ đếm TL0 được điều khiển bởi các bit C/T, GATE , TR0 /INT0 và khi đếm tràn nó thiết lập cờ ngắt TF0. Bộ đếm TH0 chỉ bị điều khiển bởi bit TR1 và khi đếm tràn nó thiết lập cờ ngắt TF1. Vậy TH0 điều khiển ngắt cho "thời gian / bộ đếm 1". "Thời gian / bộ đếm 1" trong chế độ 3 chỉ chứa chức năng đếm của nó, kết quả giống khi đặt TR1 = 0.



Hình 9.10. Điều khiển chế độ cho Timer Mode0



Hình 9.11. Điều khiển chế độ cho Timer Mode2



Hình 9.12. Điều khiển chế độ cho Timer Mode3

9.4. CƠ CHẾ NGẮT CỦA HỆ VI XỬ LÝ ON-CHIP 80C51

9.4.1. Phân loại ngắt trong hệ vi xử lý on-chip

Hệ vi xử lý on-chip 80C51 có 6 ngắt gồm 2 ngắt ngoài là /INT0, /INT1 và 4 ngắt trong, trong đó 2 ngắt của khối "thời gian / bộ đếm" và 2 ngắt phục vụ cổng truyền tin nối tiếp.

- Các ngắt ngoài có thể kích hoạt theo mức hoặc kích hoạt theo sườn xung tuỳ thuộc vào gia trị các bit của IT0, IT1 trong thanh ghi <TCON>.

Ngắt ngoài có 2 cờ ngắt tương ứng với 2 ngắt là IE0, IE1 cũng nằm trong thanh ghi <TCON>. Khi một ngắt được thực hiện thì cờ ngắt tương ứng của nó bị xoá bằng phần cứng.

♦ Các ngắt trong, với ngắt "thời gian / bộ đếm" 0 và 1 được phát sinh bởi cờ ngắt TF0, TF1. Hai cờ ngắt này được thiết lập khi thanh ghi thời gian/bộ đếm thực hiện quay vòng. Khi một ngắt được thực hiện thì cờ ngắt tương ứng phát sinh ra ngắt sẽ bị xoá bằng phần cứng trong on-chip. Các ngắt cổng nối tiếp được phát sinh bởi các ngắt R1, T1 thông qua phần tử logic OR, khi chương trình con phục vụ ngắt được kích hoạt thì các cờ ngắt phát sinh tương ứng được xoá bằng phần mềm. Các ngắt trong có thể được phép hoặc không được phép kích hoạt bằng cách đặt bằng 1 hoặc xoá về 0 cho một bit trong thanh ghi IE của <SFR>. Cấu trúc và chức năng các bit của thanh ghi cho phép ngắt IE như được thể hiện trên hình 9.13.

MSB							LSB
EA	-	-	ES	ET1	EX1	ET0	EX0

Bit	Kí hiệu	Chức năng
7	EA	Bit cho phép ngắt hoạt động , khi EA=0 thì không có ngắt nào khoạt động, khi EA=1 sẽ cho phép ngắt hoạt động.
6	-	Bit dự trữ , dành cho người sử dụng định nghĩa.
5	-	Bit dự trữ , dành cho người sử dụng định nghĩa.
4	ES	Bit cho phép hoặc không cho phép ngắt cổng nối tiếp, được đặt xoá bằng phần mềm.
3	ET1	Bit cho phép hoặc không cho phép ngắt tràn 'thời gian / bộ đếm 1'.
2	EX1	Bit cho phép hoặc không cho phép ngắt ngoài 1.
1	ET0	Bit cho phép hoặc không cho phép ngắt tràn 'thời gian / bộ đếm 0' .
0	EX0	Bit cho phép hoặc không cho phép ngắt ngoài 0

9.4.2. Mức ngắt ưu tiên trong hệ vi xử lý on-chip

Mỗi ngắt có thể lập trình riêng cho 1 hoặc 2 mức ưu tiên bằng cách đặt hoặc xoá bit tương ứng trong thanh ghi mức ưu tiên ngắt <IP> (Interrupt Priority) của <SFR>. Mỗi ngắt ưu tiên ở mức thấp có thể được ngắt bằng ngắt ưu tiên ở mức cao hơn nhưng không thể ngắt bằng ngắt có mức ưu tiên ở mức thấp hơn được. Nếu có yêu cầu ngắt của 2 mức ưu tiên cùng nhau, nếu ngắt nào có mức ưu tiên cao hơn sẽ được phục vụ. Nếu nhận được các ngắt cùng mức ưu tiên thì thứ tự quay vòng bên trong sẽ quyết định ngắt nào được phục vụ. Cấu trúc và chức năng các bit của thanh ghi ưu tiên ngắt <IP> như được thể hiện trên hình 9.14.

Thứ tự ưu tiên ngắt từ cao xuống thấp của on-chip như sau: Ngắt IE0, TF0, IE1, TF1, và R1 T1 là 2 ngắt có cùng mức ưu tiên thấp nhất.

MSB LSB

			PS	PT1	PX1	PT0	PX0

Bit	Kí hiệu	Chức năng
7	-	Bit dự trữ , dành cho người sử dụng định nghĩa.
6	-	Bit dự trữ , dành cho người sử dụng định nghĩa.
5	-	Bit dự trữ , dành cho người sử dụng định nghĩa.
4	PS	Định nghĩa mức ưu tiên của ngắt cổng nối tiếp, PS = 1 mức ưu tiên cao.
3	PT1	Định nghĩa mức ưu tiên của ngắt thời gian / bộ đếm 1, PT1=1 mức ưu tiên cao.
2	PX1	Định nghĩa mức ưu tiên của ngắt ngắt ngoài 1, PX1= 1 có mức ưu tiên cao.
1	PT0	Bit lập hoặc xoá mức ưu tiên của ngắt thời gian / bộ đếm 0
0	PX0	Định nghĩa mức ưu tiên của ngắt ngắt ngoài 0, PX0 = 1 có mức ưu tiên cao.

Hình 9.14. **Thanh ghi ưu tiên ngắt <IP>**

9.4.3. Nguyên lý điều khiển ngắt của hệ vi xử lý on-chip

Các cờ ngắt được lập ở thời điểm P2-S5 của mỗi chu kỳ máy. Chu kỳ máy tiếp theo sau chu kỳ máy có cờ ngắt được thiết lập thì chương trình con được thực hiện khi có lệnh gọi <LCALL>. Lệnh <LCALL> phát sinh nhưng lại bị cấm thực hiện khi gặp các tình huống sau:

- Đồng thời có ngắt với mức ưu tiên cao hơn hoặc bằng ngắt đang phục vụ.
- Chu kỳ máy hiện hành không phải là chu kỳ máy cuối cùng đang thực hiện của lệnh hiện hành.
- Lệnh đang thực hiện là lệnh <RET> hoặc bất cứ lệnh nào ghi vào thanh ghi <IE > hoặc <IP>.

Bất kỳ một trong 3 điều kiện này xuất hiện sẽ cản trở việc tạo ra LCALL đối với chương trình phục vụ ngắt. Điều kiện 2 đảm bảo rằng lệnh đang được thực hiện sẽ được hoàn thành trước khi trỏ tới bất kỳ chương trình phục vụ nào. Điều kiện 3 đảm bảo rằng nếu lệnh đang được thực hiện là RETI hoặc bất kỳ truy cập nào vào IE hoặc IP, thì ít nhất một lệnh nữa sẽ được thực hiện trước khi véc tơ ngắt có hiệu lực. Chu trình kiểm tra vòng được lặp lại với mỗi chu trình máy, và các giá trị được kiểm tra là các giá trị mà đã xuất hiện ở giai đoạn S5-P2 của chu trình máy trước đó. Lưu ý rằng nếu một chỉ thị ngắt có hiệu lực

nhưng không được đáp ứng đối với một trong các điều kiện trên và nếu chỉ thị này vẫn chưa có hiệu lực khi điều kiện cản trở được loại bỏ, thì ngắt bị từ chối nàysẽ không được phục vụ nữa.

LCALL do phần cứng tạo ra sẽ chuyển nội dung của bộ đếm chương trình vào ngăn xếp (nhưng không lưu lại PSW) và nạp cho PC một địa chỉ phụ thuộc vào nguồn gây ngắt như dưới đây:

Nguồn ngắt	Địa chỉ Véc tơ ngắt
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H

Lệnh RETI thông báo cho bộ xử lý rằng thủ tục ngắt này đã kết thúc, sau đó lấy ra 2 byte đầu tiên từ ngăn xếp và tái nạp vào bộ đếm chương trình để trả lại quyền điều khiển cho chương trình chính.

Vì các chốt ngắt ngoài được trích mẫu một lần trong mỗi chu trình máy, nên một giá trị cao hoặc thấp của đầu vào sẽ duy trì trong ít nhất là 12 chu kỳ nhịp của bộ dao động để đảm bảo giữ được mẫu. Nếu ngắt ngoài được kích hoạt bằng sườn xung, thì nguồn ngắt ngoài phải duy trì ở chốt yêu cầu giá trị cao ít nhất 1 chu kỳ máy và sau đó duy trì giá trị thấp ít nhất 1 chu kỳ máy nữa. Việc này được thực hiện để đảm bảo rằng quá trình chuyển tiếp cho thấy là chỉ thị yêu cầu ngắt IEx sẽ được xác lập. IEx sẽ tự động được xoá bởi CPU khi thủ tục ngắt đáp ứng được gọi.

Nếu ngắt ngoài được kích hoạt theo mức, thì nguồn ngắt bên ngoài phải duy trì cho yêu cầu này có hiệu lực cho đến khi thực sự tạo ra ngắt đã được yêu cầu. Sau đó nó phải huỷ yêu cầu đó trước khi thủ tục phục vụ ngắt hoàn thành vì nếu không sẽ tạo ra ngắt khác.

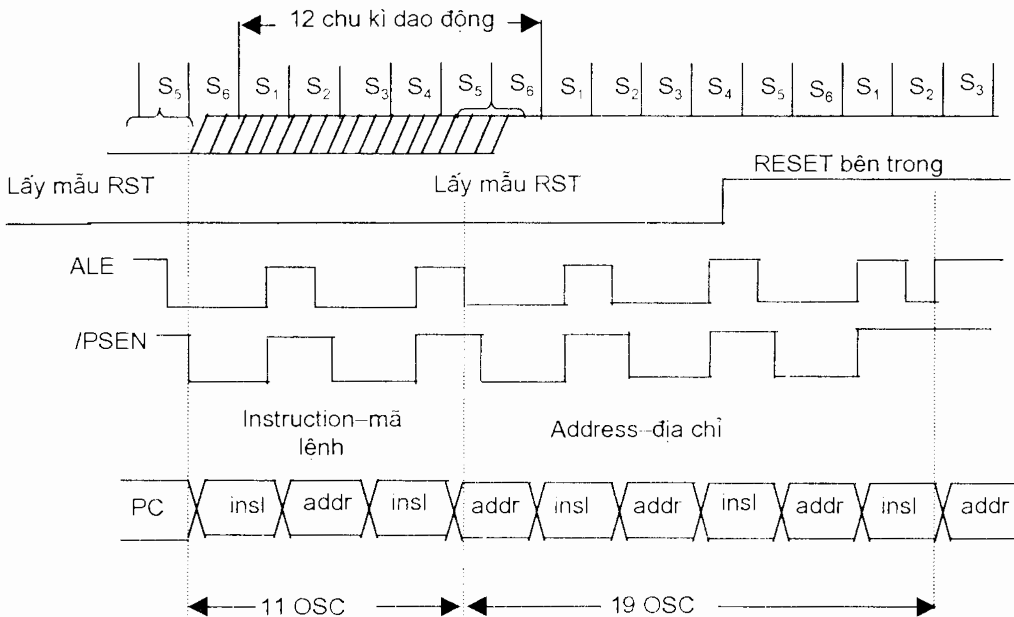
Cấu trúc ngắt của hệ vi xử lý on-chip 80C51 cho phép thực hiện theo các bước đơn với sự tham gia rất ít của phần mềm. Như đã lưu ý trước đây, một yêu cầu ngắt sẽ không được đáp ứng trong khi một ngắt có mức độ ưu tiên tương đương vẫn đang hoạt động, nó cũng không được đáp ứng sau khi có lệnh RETI cho đến khi có ít nhất một lệnh khác đã được thực hiện. Do đó mỗi khi một thủ tục ngắt được đưa vào, thì nó không thể được đưa vào lần nữa cho đến khi ít nhất một lệnh của chương trình ngắt được thực hiện. Một cách để sử dụng đặc điểm này đối với hoạt động theo bước đơn lẻ là lập trình cho một trong những ngắt ngoài (chẳng hạn INT0) được kích hoạt theo mức. Thủ tục đáp ứng cho ngắt sẽ kết thúc bằng mã dưới đây:

```
JNB     P3.2,S     ; Đợi đến khi INT0 = 1.
JB      P3.2,S     ; Đợi đến khi INT0 = 0.
RETI               ; Trở lại và thực hiện một lệnh.
```

Bây giờ nếu chốt INT0 (cũng là chốt của P3.2) được duy trì ở mức thấp, thì CPU sẽ chuyển ngay đến thủ tục ngắt ngoài 0 (IE0) và dừng ở đó cho đến khi INT0 nhận xung từ thấp lên cao rồi xuống thấp. Sau đó nó sẽ thực hiện lệnh RETI, trở lại nhiệm vụ chương trình, thực hiện một lệnh, và ngay sau đó nhập lại thủ tục ngắt ngoài 0 (IE0) để đợi xung nhịp tiếp theo của P3.2.

9.4.4. Nguyên lý khởi động của on-chip 80C51

Nguyên lý khởi động lại



***Hình 9.15**. Khởi động lại hệ VXL on-chip*

Tín hiệu khởi động lại on-chip được đưa vào qua chân RST (đây là tín hiệu của trigơ Schmit) bằng cách giữ chân RST ở mức logic cao trong khoảng 2 chu kỳ máy (24 chu kỳ dao động). Khi đó on-chip sẽ được khởi động lại theo đồ thị thời gian như thể hiện trên hình 9.15.

Tín hiệu khởi động lại bên ngoài đưa vào chân RST không đồng bộ với thời gian RESET bên trong. Chân RST sẽ được lấy mẫu tại S5-P2 của mỗi chu kỳ máy, các chân của cổng sẽ giữ hoạt động hiện hành của chúng cho 19 chu kỳ dao động sau khi giá trị logic 1 đã được lấy mẫu ở chân RST. Sau khi khởi động lại giá trị của các thanh ghi chức năng đặc biệt <SFR> được gán giá trị như hình 9.16.

Thanh ghi của SFR	Gía trị sau RESET
PC	0000h
Acc	00h
B	00h
PSW	00h
SP	07h
DPTR	0000h
P0-:-P3	FFh
IP	XXX00000b
IE	0XX00000b
TMOD	00h
TCON	00h
TH0	00h
TL0	00h
TH1	00h
TL1	00h
SCON	00h
SBUF	
PCON(NMOS)	0xxxxxxxb
PCON(CMOS)	0xxx0000b

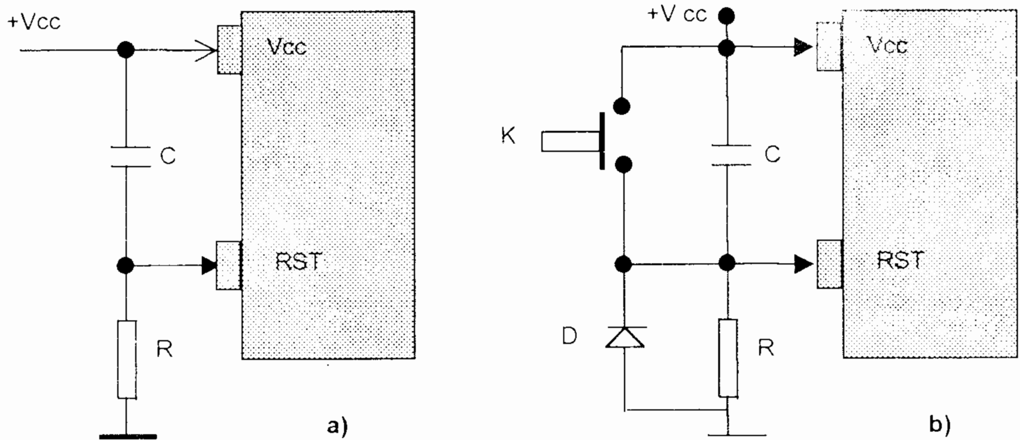
***Hình 9.16.* Giá trị của các thanh ghi sau khi Reset**

Sau khi Reset có hiệu lực các chốt cổng được gán giá trị FFh, con trỏ ngăn xếp có giá trị 07h và thanh ghi <SBUF> có giá trị bất định, các thanh ghi <PCON>, <IP>.<IE> có một số bit có giá trị bất định, các thanh ghi còn lại có giá trị 00h.

Riêng đối với RAM bên trong on-chip 80C51 khi cấp nguồn nuôi hay RESET lại không bị tác động, mà nội dung trong RAM có giá trị ngẫu nhiên.

Khởi động lại cho on-chip có thể hoạt động ở trạng thái tự động hoặc trạng thái bán tự động như thể hiện trên hình 9.17.

Khởi động tự động có thể được tạo ra khi cấp nguồn điện <+Vcc> cho on-chip bằng mạch điện RC. Sau khi cấp nguồn, mạch điện RC giữ cho chân RST ở trạng thái cao trong thời gian tuỳ thuộc vào hằng số thời gian của mạch RC. Để bảo đảm khởi động được On-chip, thời gian chân RST ở trạng thái cao phải đủ lớn (khoảng lớn hơn 2 chu kỳ máy) để mạch dao động chuyển sang trạng thái dao động ổn định. Khởi động bán tự động như hình<b>, sau khi cấp nguồn điện mạch sẽ tự động RESET như hình <a>. Khi cần khởi động lại phải nhấn công tắc thường ngắt K. Thời gian nhấn công tắc chân RST phải ở trạng thái cao.



Hình 9.17. Mạch RESET on-chip 89C51

9.5. NGUYÊN LÝ TRUYỀN TIN NỐI TIẾP CỦA HỆ VI XỬ LÝ ON-CHIP 80C51

On-chip 80C51 truyền tin nối tiếp bằng cổng RxT (P3.0) và TxD (P3.1), dữ liệu xuất nhập truyền qua cổng nối tiếp bằng tốc độ baud và đều qua bộ đệm nối tiếp <SBUF>. Cổng nối tiếp của on-chip là cổng truyền tin hai chiều với khả năng vừa thực hiện chức năng nhận vừa thực hiện chức năng đệm, tức là nó có thể nhận byte kế tiếp trước khi byte được nhận trước đó được đọc từ thanh ghi đệm. (Tuy nhiên, nếu byte đầu tiên vẫn chưa được đọc tại thời điểm nhận của byte thứ hai, thì byte này sẽ bị mất). Điều khiển cổng nối tiếp bằng thanh ghi <SCON>, trạng thái của 2 bit <SM0, SM1> trong <SBUF> thiết lập lên 4 chế độ hoạt động giao tiếp nối tiếp chuẩn.

- *Chế độ 0:* Dữ liệu nối tiếp vào và ra sẽ thông qua chân RxD. Chân TxD sẽ đưa ra xung nhịp đồng hồ. Khung dữ liệu gồm 8 bit được truyền-nhận nối tiếp (đầu tiên là bit có trọng số thấp nhất LSB). Tốc độ baud được cố định bằng 1/12 tần số của bộ dao động.
- *Chế độ 1:* ở chế độ này khung dữ liệu có 10 bit được truyền (thông qua TxD) hoặc được nhận (qua RxD) gồm bit khởi đầu (có giá trị 0), 8 bit dữ liệu (đầu tiên là LSB) và một bit dừng STOP (có giá trị 1). Khi nhận, bit dừng chuyển vào RB8 ở thanh ghi chức năng đặc biệt (SCON). Tốc độ baud có thể thay đổi được trong chế độ này.
- *Chế độ 2:* ở chế độ này 11 bit được truyền thông qua TxD hoặc được nhận thông qua RxD gồm bit khởi đầu (0), 8 bit dữ liệu (đầu tiên là LSB), 1 bit dữ liệu thứ 9 có thể lập trình được, và một bit dừng (1). Khi truyền thì bit dữ liệu thứ 9 (TB8 ở SCON) có thể được gán giá trị 0

hoặc 1. Chẳng hạn như bit chẵn lẻ PARITY (P trong PSW) có thể được chuyển vào TB8. Khi nhận thì bit dữ liệu thứ 9 chuyển vào RB8 ở thanh ghi chức năng đặc biệt SCON, trong khi bit dừng STOP được lọc bỏ. Tốc độ baud có thể lập trình được bằng 1/32 hoặc 1/64 tần số bộ dao động.

- *Chế độ 3:* ở chế độ này thì 11 bit được truyền (thông qua TxD) hoặc được nhận (thông qua RxD), đó là bit khởi đầu START(0), 8 bit dữ liệu (đầu tiên là LSB), 1 bit dữ liệu thứ 9 có thể lập trình được, và 1 bit dừng STOP(1). Trên thực tế chế độ 3 giống chế độ 2 ở mọi góc độ trừ tốc độ baud. Tốc độ baud ở chế độ 3 có thể thay đổi và được xác định theo bộ thời gian Timer1.

Trong cả 4 chế độ, thì việc truyền được bắt đầu bởi lệnh bất kỳ nào mà sử dụng thanh ghi SBUF như một thanh ghi đích. Việc nhận cũng được bắt đầu ở các chế độ khi bit cho phép máy thu REN = 1.

Các chế độ 2 và 3 có một dự trù đặc biệt cho các liên lạc đa xử lý. Trong chế độ này thì 9 bit dữ liệu được sử dụng (chứ không phải 8 bit). Bit thứ 9 sẽ chuyển vào RB8, sau đó là 1 bit dừng. Cổng nối tiếp có thể được lập trình thoả mãn điều kiện khi bit dừng được nhận thì ngắt của cổng nối tiếp sẽ được kích hoạt khi RB8= 1. Đặc điểm này có thể thực hiện được bằng việc đưa bit SM2 vào SCON.

Khi bộ xử lý chủ (MASTER) muốn truyền 1 khối dữ liệu đến một trong những bộ xử lý khác (SLAVE), thì đầu tiên nó gửi đi một byte địa chỉ xác định của bộ xử lý đích. Nội dung byte địa chỉ khác với nội dung byte dữ liệu ở chỗ là bit thứ 9 bằng 1 ở byte địa chỉ và bằng 0 ở byte dữ liệu. Với SM2 = 1 được gán, thì không có bộ xử lý nào bị ngắt bởi 1 byte dữ liệu. Tuy nhiên một byte địa chỉ sẽ ngắt tất cả các bộ xử lý thành phần, để cho mỗi bộ xử lý thành phần có thể kiểm tra byte nhận được và để xem có phải nó được qui chiếu tới không. Bộ xử lý nào được trỏ tới sẽ xoá bit SM2 của nó và chuẩn bị nhận các byte dữ liệu sẽ đưa đến ở bước sau. Các bộ xử lý khác mà không được qui chiếu sẽ tiếp tục hoạt động của mình mà không cần quan tâm tới dữ liệu trên kênh.

Trong chế độ 0 thì SM2 không có tác dụng còn ở chế độ 1 nó có thể được sử dụng để kiểm tra bit dừng STOP. Trong quá trình nhận tin ở chế độ 1, nếu SM2 =1, thì ngắt khi nhận tin sẽ được kích hoạt chỉ khi bit dừng được nhận vào.

Thanh ghi điều khiển cổng nối tiếp

Thanh ghi trạng thái và điều khiển cổng nối tiếp là một thanh ghi chức năng đặc biệt SCON, minh hoạ trên hình 9.18. Thanh ghi này không những chứa các bit chọn chế độ mà còn chứa bit dữ liệu thứ 9 dành cho việc truyền và nhận (TB8 và RB8), và chứa các bit ngắt cổng nối tiếp (T1 và R1).

MSB LSB

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Trong đó SM0, SM1 xác định chế độ của cổng nối tiếp như sau:

SM0	SM1	Chế độ	Đặc điểm	Tốc độ baud
0	0	0	Thanh ghi dịch	fosc/12
0	1	1	UART 8 bít	Có thể thay đổi
1	0	2	UART 9 bít	fosc/64 hoặc fosc/32
1	1	3	UART 9 bít	Có thể thay đổi

SM2 Cho phép liên lạc đa xử lý thể hiện ở chế độ 2 và 3. Trong chế độ 2 hoặc 3, nếu SM2 = 1, thì R1 sẽ không được kích hoạt nếu bít dữ liệu thứ 9 (RB8) nhận được bằng không. Trong chế độ 1, nếu SM2 = 1 thì R1 sẽ không được kích hoạt nếu bít dừng STOP không nhận được. Trong chế độ 0, SM2 nên gán bằng 0.

REN Cho phép máy thu nối tiếp làm việc nếu REN=1. Nếu REN=0 máy thu nối tiếp sẽ bị cấm hoạt động.

TB8 Là bit dữ liệu thứ 9 mà sẽ được truyền ở chế độ 2 và 3. Được lập và xoá bởi chương trình khi cần.

RB8 Là bit dữ liệu thứ 9 của tuyến thu ở chế độ 2 và 3. Trong chế độ 1, nếu SM2 = 0, thì RB8 là bít dừng STOP đã thu được.Trong chế độ 0, RB8 không được sử dụng.

TI Cờ ngắt tuyến phát. Cờ này do phần cứng thiết lập ở cuối thời điểm của bit thứ 8 trong chế độ 0, hoặc ở đầu thời điểm của bit dừng trong các chế độ khác. Cờ này bắt buộc phải được xoá bằng phần mềm.

RI Cờ ngắt tuyến thu. Cờ này do phần cứng xác lập ở thời điểm cuối của bít thứ 8 trong chế độ 0, hoặc ở giữa thời điểm của bít dừng trong các chế độ khác (trừ trường hợp ngoại lệ, xem SM2). Cờ này bắt buộc phải được xoá bằng phần mềm.

Hình 9.18. **Thanh ghi điều khiển cổng nối tiếp Serial Controller (SCON)**

Các tốc độ baud

Tốc độ baud ở chế độ 0 được cố định: *Tốc độ baud chế độ 0 = tần số bộ dao động / 12.*

Tốc độ baud ở chế độ 2 phụ thuộc vào giá trị của bit SMOD trong thanh ghi chức năng đặc biệt PCON. Nếu SMOD = 0 (Tức là giá trị khi xác lập lại), thì tốc độ baud là 1/64 tần số của bộ dao động. Nếu SMOD = 1, thì tốc độ baud bằng 1/32 tần số bộ dao động.

$$\text{Tốc độ baud chế độ 2} = \frac{2^{SMOD}}{64} \times (\text{Tần số bộ dao động})$$

Trong 80C51, các tốc độ baud ở chế độ 1 và 3 do Timer 1 quyết định.

Sử dụng Timer 1 để tạo ra các tốc độ baud

Khi Timer 1 được sử dụng như bộ tạo tốc độ baud thì các tốc độ baud ở chế độ 1 và 3 do tốc độ tràn của Timer 1 và giá trị của SMOD quyết định:

$$\text{Tốc độ baud ở chế độ 1, 3} = \frac{2^{SMOD}}{32} \times (\text{Tốc độ tràn của Timer 1})$$

Ngắt của Timer 1 sẽ mất tác dụng trong ứng dụng này.

Ta có thể nhận được các tốc độ baud rất thấp với Timer 1 bằng cách làm cho ngắt của Timer 1 có tác dụng, và thiết lập Timer 1 để hoạt động như một bộ đếm thời gian 16 bít (4 bít cao của TMOD = 0001B). Hình 9.19 liệt kê các tốc độ baud khác nhau thường được sử dụng và cách chúng có thể nhận được từ Timer 1.

Tốc độ Baud	Tần số (MHz)	SMOD	Timer 1		
			C/T	Chế độ	Giá trị tái nạp
Cực đại chế độ 0: 1 MHz	12	×	×	×	×
Cực đại chế độ 2: 375 K	12	1	×	×	×
Chế độ 1, 3: 62.5 K	12	1	0	2	FFH
19.2 K	11.059	1	0	2	FDH
9.6 K	11.059	0	0	2	FDH
4.8 K	11.059	0	0	2	FAH
2.4 K	11.059	0	0	2	F4H
1.2 K	11.059	0	0	2	E8H
137.5	11.986	0	0	2	1DH
110	6	0	0	2	72H
110	12	0	0	1	FEEBH

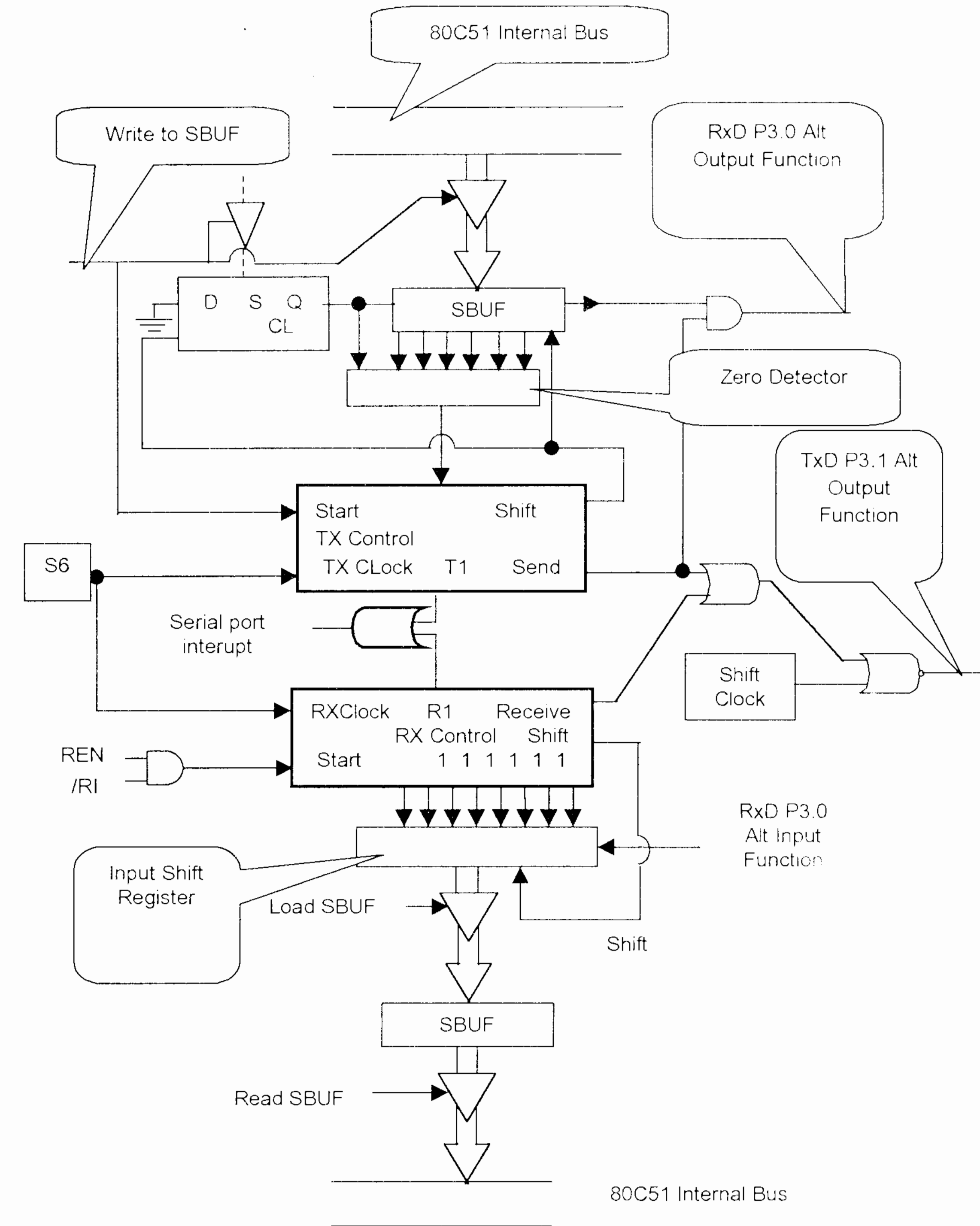
***Hình 9.19.* Các tốc độ baud thường dùng do Timer 1 tạo ra**

Hoạt động của chế độ 0

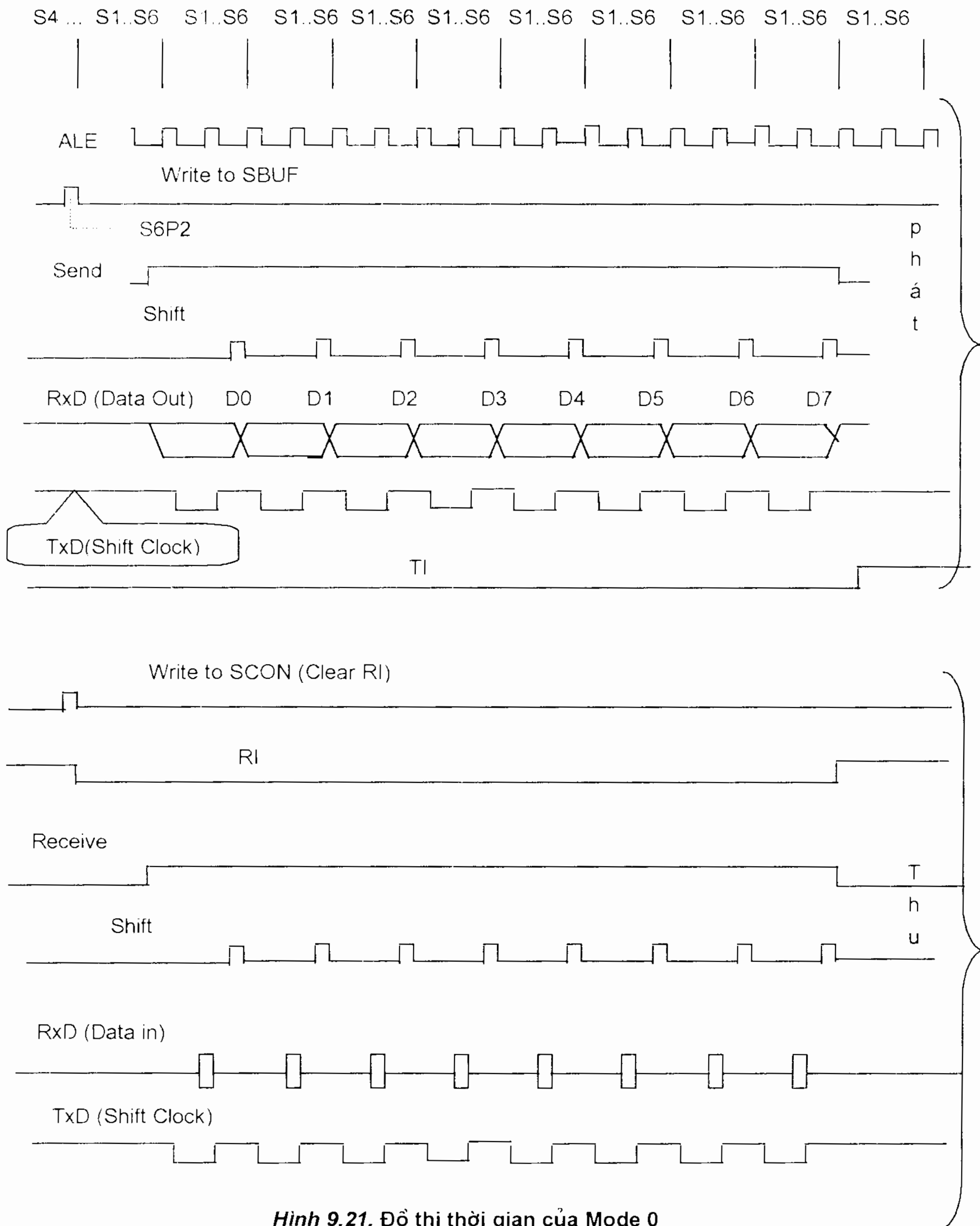
Dữ liệu nối tiếp vào và ra thông qua RxD. TxD cho ra đồng hồ xung nhịp. Khung dữ liệu 8 bit được truyền - nhận (LSB đầu tiên). Tốc độ baud được cố định bằng 1/12 tần số bộ dao động.

Hình 9.20 mô tả sơ đồ chức năng của cổng nối tiếp ở chế độ 0 và các điểm dấu thời gian có liên quan. Quá trình truyền được khởi đầu bằng bất kỳ lệnh nào mà sử dụng đến SBUF như một thanh ghi đích. Tín hiệu "ghi vào SBUF" ở giai đoạn S6P2 cũng nạp giá trị 1 vào vị trí thứ 9 của thanh ghi dịch truyền và

bật cờ báo cho khối điều khiển phát về yêu cầu truyền tin. Thời gian được xác lập bên trong cho một chu trình máy đầy đủ sẽ bắt đầu từ thời điểm "ghi vào SBUF"cho tới khi SEND được kích hoạt.



Hình 9.20. **Cổng truyền tin nối tiếp ở mode 0**



Hình 9.21. **Đồ thị thời gian của Mode 0**

SEND cho phép nội dung của thanh ghi dịch đưa ra đầu ra P3.0 và cho phép tín hiệu SHIFT CLOCK đến đầu ra P3.1. SHIFT CLOCK có giá trị thấp trong các trạng thái S3,S4 và S5 của mỗi chu kỳ máy, và có giá trị cao trong các trạng thái S6, S1 và S2. Tại trạng thái S6P2 của mỗi chu kỳ máy khi SEND có mức tích cực, thì nội dung của thanh ghi dịch phát được dịch sang bên phải một bit.

Khi các bit dữ liệu dịch sang bên phải thì các giá trị 0 được gán vào bên trái. Khi bit có trọng số lớn nhất MSB của byte dữ liệu ở vị trí đầu ra của thanh ghi dịch thì giá trị 1 (đã được nạp từ đầu vào vị trí thứ 9) được đặt vào bên trái của MSB, và tất cả các vị trí ở bên trái của MSB đều chứa các giá trị 0. Điều kiện này sẽ chỉ thị cho khối điều khiển TX thực hiện một phép dịch cuối cùng và sau đó huỷ tác dụng của SEND và lập cờ ngắt TI. Cả hai tác động này xảy ra ở trạng thái S1P1 của chu trình máy thứ 10 kể từ thời điểm "ghi vào SBUF".

Quá trình nhận tin được khởi đầu bằng điều kiện REN = 1 và RI = 0. Trong trạng thái S6P2 của chu kỳ máy tiếp theo, khối điều khiển RX sẽ ghi các bít 11111110 vào thanh ghi dịch nhận, và trong pha xung nhịp tiếp theo sẽ kích hoạt RECEIVE .

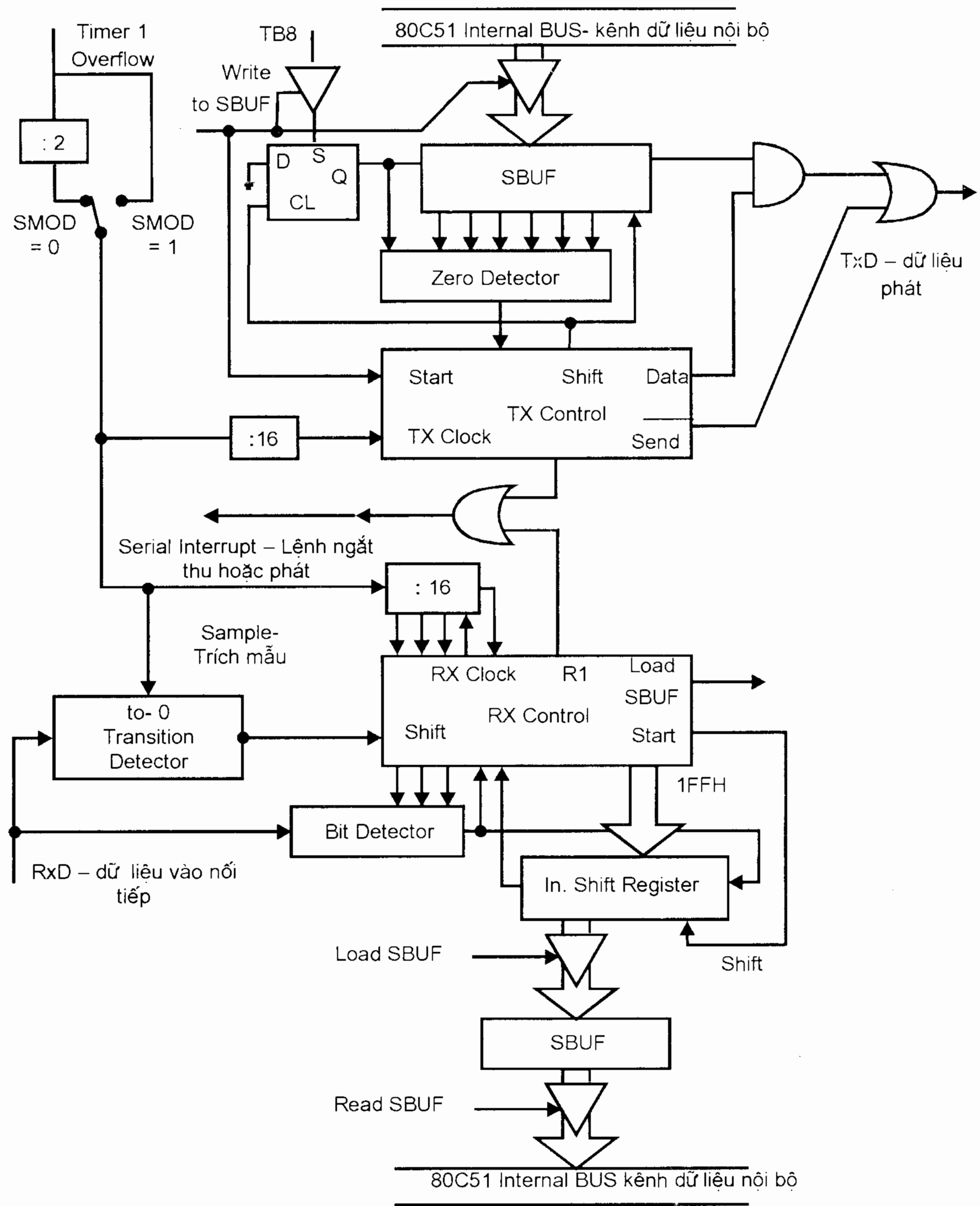
Về phần mình, RECEIVE cho phép SHIFT CLOCK (đồng hồ xung nhịp) đưa đến đầu ra P3.1. SHIFT CLOCK sẽ tạo ra việc phát tin tại giai đoạn S3P1 và S6P1 của mỗi chu trình máy. Tại giai đoạn S6P2 của mỗi chu trình máy khi RECEIVE có mức tích cực thì nội dung của thanh ghi dịch nhận tin được dịch sang trái một vị trí. Giá trị đưa vào từ bên phải là giá trị đã được tạo mẫu ở chân P3.0 ở giai đoạn S5P2 của cùng chu trình máy.

Khi các bit dữ liệu được đưa vào từ bên phải, thì các giá trị 1 sẽ đi ra bên trái. Khi giá trị 0 (đã được nạp ban đầu vào vị trí tận cùng bên phải) dịch đến vị trí tận cùng bên trái trong thanh ghi dịch, thì nó chỉ thị cho khối điều khiển RX thực hiện phép dịch cuối cùng và nạp vào bộ đệm SBUF. Tại giai đoạn S1P1 của chu trình máy thứ 10 sau thời điểm ghi vào SCON (đã xoá RI), thì RECEIVE được xoá khi RI được xác lập.

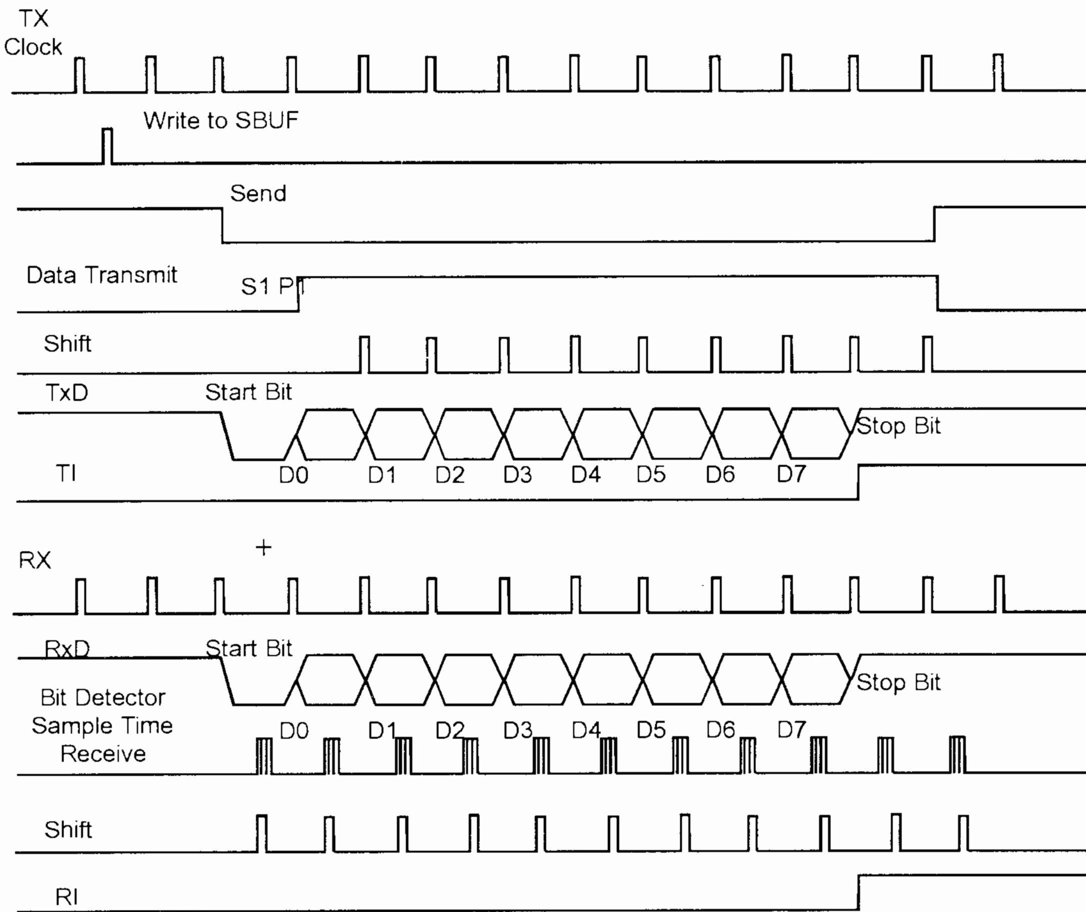
Hoạt động của chế độ 1

Ở chế độ này, 10 bit được truyền (qua TxD), hoặc được nhận (qua RxD), đó là 1 bit khởi đầu START (0), 8 bit dữ liệu DATA (LSB đầu tiên) và 1 bit dừng STOP (1). Khi nhận tin, bit dừng STOP chuyển vào RB8 trong SCON. Hình 9.22 là sơ đồ chức năng của cổng nối tiếp ở chế độ 1 và hình 9.23 là đồ thị thời gian liên quan tới quá trình thu-phát thông tin của chế độ này.

Quá trình truyền được khởi đầu bởi lệnh bất kỳ có sử dụng SBUF như một thanh ghi đích. Tín hiệu "ghi vào SBUF" sẽ nạp giá trị 1 vào vị trí bit thứ 9 của thanh ghi dịch truyền và báo cho khối điều khiển TX về yêu cầu cần truyền tin. Quá trình truyền thực tế bắt đầu ở giai đoạn S1P1 của chu kỳ máy theo sau chu kỳ kế tiếp của bộ đếm 16. (Do đó, các thời điểm của bit truyền được đồng bộ với nhịp bộ đếm 16, chứ không phải với tín hiệu "ghi vào SBUF").



Hình 9.22. **Cổng truyền tin nối tiếp Mode 1**



Hình 9.23. **Đồ thị thời gian của Mode 1**

Quá trình truyền tin bắt đầu bằng sự kích hoạt của tín hiệu /SEND dùng để mở cổng OR cho TxD (đặt bit khởi đầu tại TxD). Sau đó là đến tín hiệu DATA được kích hoạt để mở tiếp cổng AND. Điều này sẽ cho phép mở thông đường truyền từ thanh ghi dịch truyền đến đầu ra TxD. Xung nhịp để dịch các bit trong thanh ghi dịch truyền sẽ xuất hiện ngay sau đó.

Khi các bt dữ liệu dịch sang phải, thì các giá trị 0 được đưa vào từ bên trái. Khi MSB của byte dữ liệu ở vị trí đầu ra của thanh ghi dịch thì giá trị 1 (ban đầu đã được nạp vào vị trí thứ 9) sẽ được điền vào ngay bên trái của bit MSB còn các bit kể từ nó sang trái đều có giá trị 0. Điều kiện này sẽ chỉ thị cho khối điều khiển TX thực hiện lần dịch cuối cùng và sau đó đưa trả /SEND về mức thụ động, đồng thời xác lập cờ ngắt TI. Thời điểm này rơi vào chu kỳ thứ 10 của bộ đếm 16 sau thời điểm ‘ Write to SBUF- ghi vào SBUB’.

Quá trình thu tin được khởi đầu bằng phát hiện sự chuyển trạng thái từ 1 đến 0 ở tuyến thu RxD. Để phát hiện chính xác, tín hiệu trên tuyến RxD được

lấy mẫu ở tốc độ gấp 16 lần tốc độ baud của đường truyền. Khi một bit được phát hiện, thì bộ đếm 16 được tái xác lập ngay và giá trị 1FFH được ghi vào thanh ghi dịch đầu vào. Việc tái thiết lập bộ đếm 16 sẽ đồng nhất thời điểm tràn của bộ đếm với các mốc thời gian của bit đang đi tới đầu thu.

Bằng cách đó mỗi bit được chia thành 16 phần thời gian bằng nhau. Tại các phần thời gian thứ 7, 8 và 9 của mỗi bit, bộ phát hiện sẽ trích mẫu của RxD. Giá trị được chấp nhận là giá trị đã có ở ít nhất là 2 trong 3 mẫu (theo tiêu chuẩn đa số). Phương pháp này được thực hiện nhằm để để chống nhiễu đường truyền. Nếu giá trị được chấp nhận đối với bit đầu tiên không phải là 0 (không phải bit START), thì các mạch thu được tái xác lập để quay trở lại chờ một đột biến từ 1 đến 0 khác.

Khi các bit dữ liệu đi vào từ phía bên phải của thanh ghi dịch, thì các giá trị 1 được dịch ra sang bên trái nó. Khi bít khởi đầu đến vị trí trái cùng của thanh ghi dịch (ở chế độ 1 nó là thanh ghi 9 bit), thì nó chỉ thị cho khối điều khiển RX thực hiện phép dịch chuyển cuối cùng rồi nạp SBUF và RB8, và xác lập RI. Tín hiệu để nạp SBUF, RB8 và để xác lập RI sẽ được tạo ra khi và chỉ khi các điều kiện sau đây được thoả mãn ở thời điểm xung nhịp cuối cùng được tạo ra:

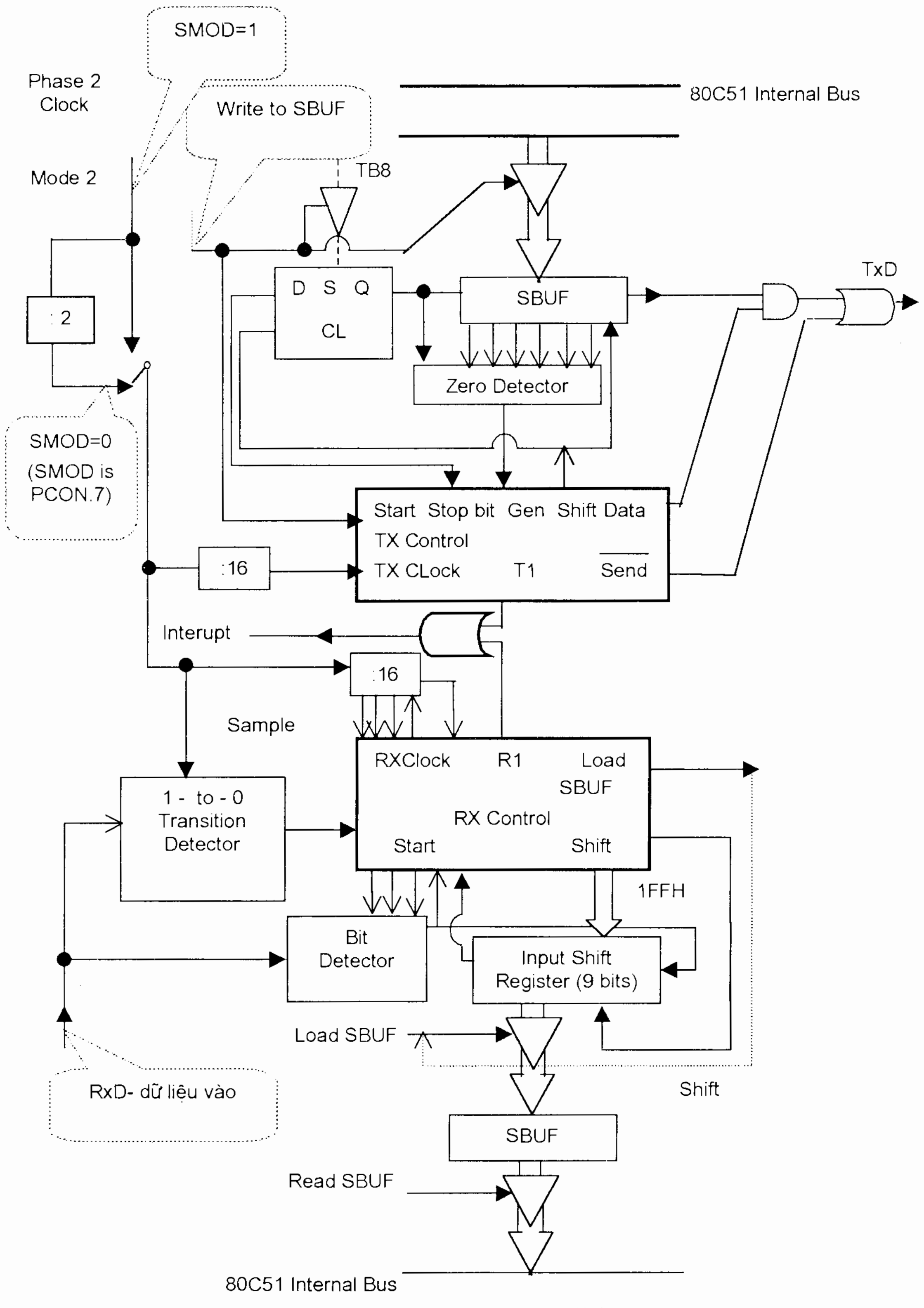
1. R1 = 0 và
2. Hoặc SM2 = 0 hoặc bit STOP nhận được = 1.

Nếu một trong hai điều kiện này không được thoả mãn, thì byte thông tin nhận được sẽ bị mất. Nếu cả hai điều kiện được thoả mãn, thì bit dừng STOP được chuyển vào RB8, 8 bit dữ liệu chuyển vào SBUF và RI được kích hoạt. Ở thời điểm này, bất kể các điều kiện trên được thoả mãn hay không, thì khối điều khiển vẫn quay trở lại để tiếp tục chức năng phát hiện đột biến mới từ 1 đến 0 trên đường thu tin RxD.

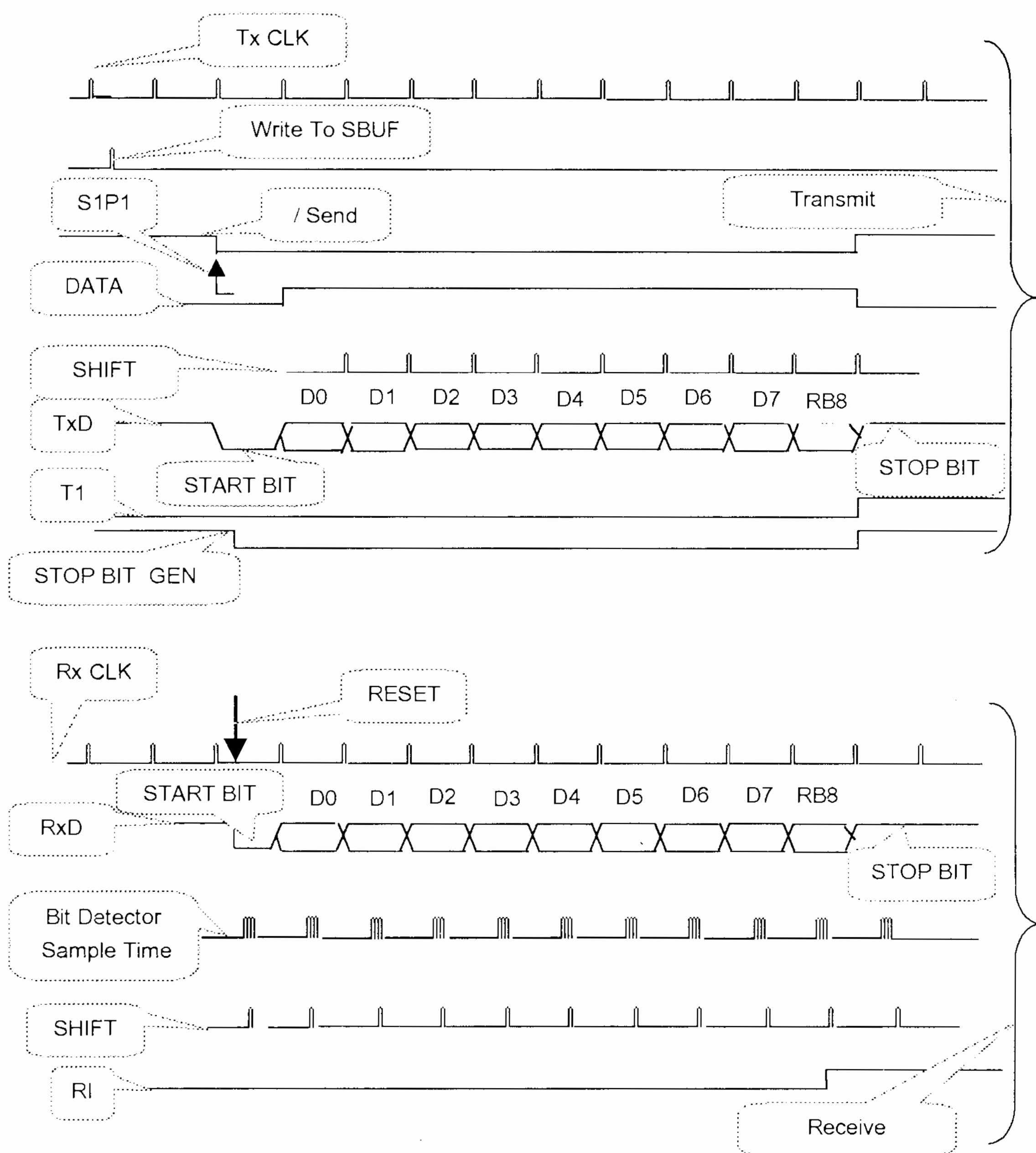
Hoạt động của chế độ 2 và 3

Ở các chế độ này có 11 bit được truyền đi (qua TxD) hoặc được nhận vào (qua RxD), đó là 1 bit khởi đầu START (0), 8 bit dữ liệu DATA (bit LSB đầu tiên), 1 bit dữ liệu thứ 9 có thể lập trình được và 1 bit dừng STOP (1). Khi truyền tin đi, bit dữ liệu thứ 9 (TB8) có thể được gán giá trị 0 hoặc 1. Khi nhận, bit dữ liệu thứ 9 chuyển vào RB8 trong SCON. Tốc độ baud có thể lập trình được bằng 1/32 hoặc 1/64 tần số của bộ giao động ở chế độ 2. Chế độ 3 có thể có tốc độ baud thay đổi do Timer 1 tạo ra.

Hình 24, 25, 26 và 27 là sơ đồ chức năng của cổng nối tiếp ở các chế độ 2 và 3 và đồ thị thời gian tương ứng của chúng. Phần nhận thông tin được tổ chức giống như chế độ 1. Phần phát tin khác với chế độ 1 chỉ ở bít thứ 9 của thanh ghi dịch truyền.



Hình 9.24. **Cổng truyền tin nối tiếp ở MODE 2**

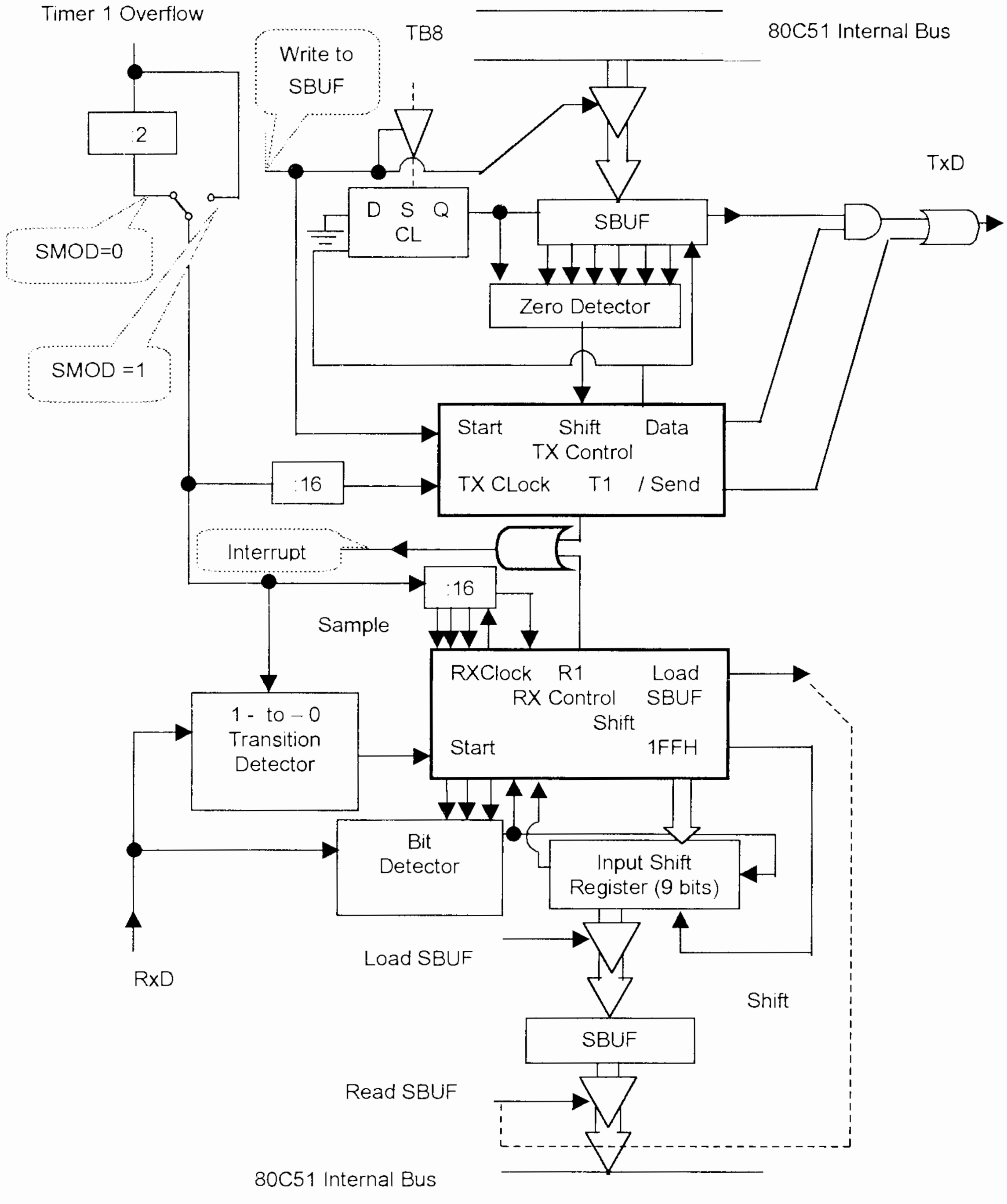


***Hình 9.25.* Đồ thị thời gian của cổng truyền tin nối tiếp ở mode 2**

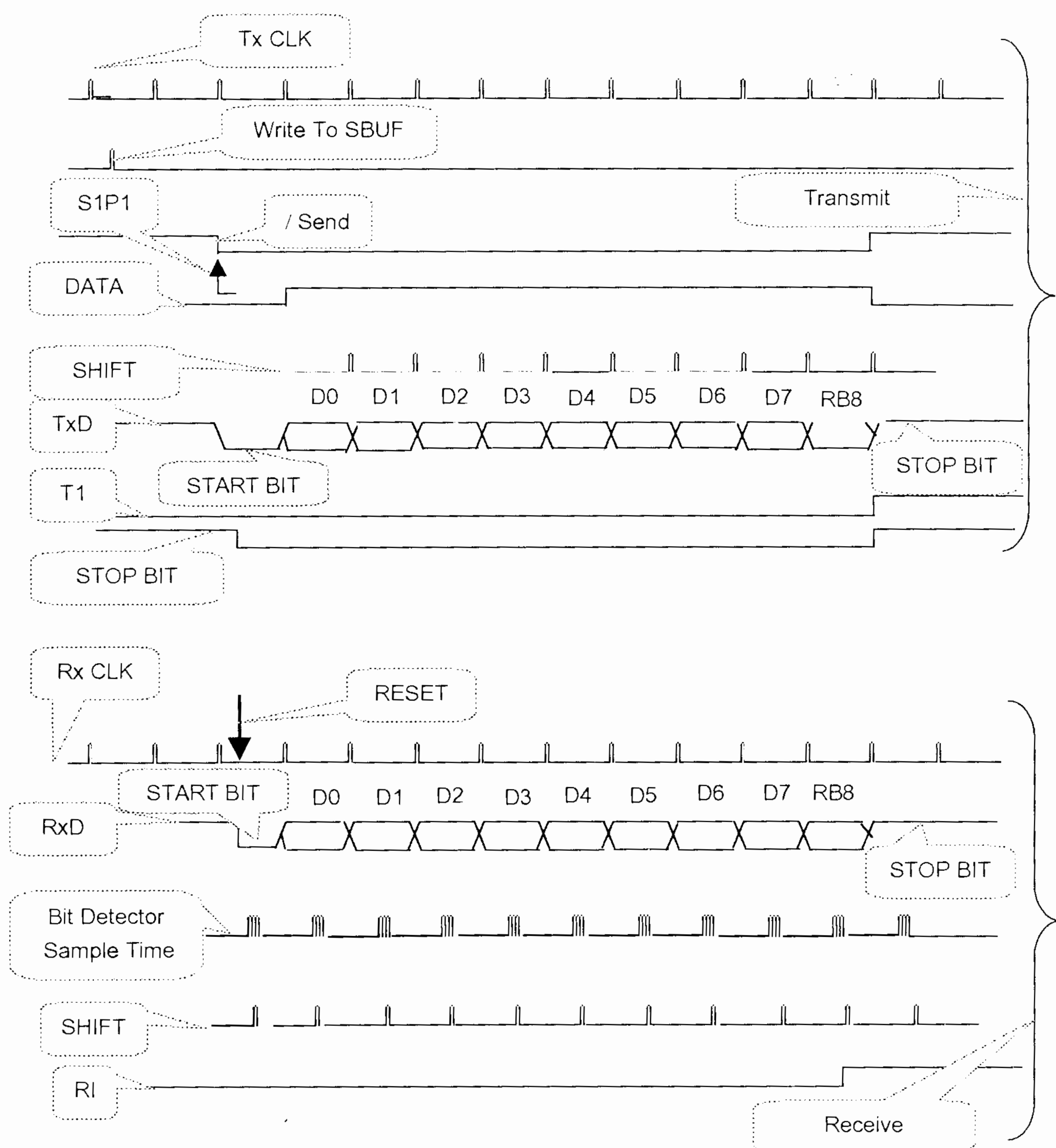
Quá trình truyền tin được khởi đầu bằng bất kỳ lệnh nào có sử dụng SBUF như một thanh ghi đích. Tín hiệu"ghi vào SBUF" cũng nạp bit TB8 vào vị trí bit thứ 9 của thanh ghi dịch truyền và chỉ thị cho khối điều khiển TX rằng có yêu cầu phải phát thông tin. Quá trình phát được bắt đầu tại trạng thái S1P1 của chu kỳ máy ngay sau thời điểm tràn của bộ đếm 16. (Do đó, các bit được đồng bộ đối với chu kỳ bộ đếm 16, chứ không phải đối với tín hiệu "ghi vào SBUF").

Quá trình phát bắt đầu bằng việc kích hoạt của tín hiệu /SEND (đặt bit khởi đầu vào TxD). Sau đó đến tín hiệu DATA được kích hoạt. Xung nhịp đầu tiên chuyển giá trị 1 (bit dừng STOP) vào vị trí bít thứ 9 của thanh ghi dịch còn

sau đó các giá trị 0 được đưa vào. Vì vậy khi các bit dữ liệu dịch ra sang phải, thì các giá trị 0 được đưa vào từ bên trái. Khi TB8 ở vị trí đầu ra của thanh ghi dịch, thì bit dừng chuyển đến bên trái của TB8, và tất cả các vị trí ở bên trái của nó đều là giá trị 0. Điều kiện này chỉ thị cho khối điều khiển TX thực hiện phép dịch cuối cùng và sau đó thụ động hoá đối với SEND và xác lập TI. Thời điểm này ứng với chu kỳ thứ 11 của bộ đếm 16 sau khi có tín hiệu "Write to SBUF".



Hình 9.26. Cổng truyền tin nối tiếp ở MODE 3



Hình 9.27. **Đồ thị thời gian của cổng truyền tin nối tiếp ở MODE 3**

Quá trình thu tin được khởi đầu bằng sự phát hiện đột biến từ 1 đến 0 ở tuyến RxD. Với mục đích đảm bảo độ tin cậy, khi trích mẫu trên RxD thì tốc độ trích mẫu được lấy gấp 16 lần tốc độ baud của đường truyền. Khi mỗi bit được trích mẫu xong, thì bộ đếm bội 16 ngay sau đó được tái xác lập và giá trị 1FFH được ghi vào thanh ghi dịch đầu vào.

Ở các phần thời gian thứ 7, 8 và 9 của mỗi bit, bộ trích mẫu sẽ đưa ra mẫu giá trị của RxD. Giá trị được chấp nhận là giá trị đã thấy ở ít nhất hai trong ba mẫu. Nếu giá trị được chấp nhận của bit đầu tiên không phải là 0, thì khối điều khiển RX trở lại dò tìm phép chuyển dịch 1 đến 0 khác. Nếu giá trị được chấp

nhận đối với bit đầu tiên không phải là 0 (không phải bit START), thì các mạch thu được tái xác lập để quay trở lại chờ một đột biến từ 1 xuống 0 khác.

Khi các bit dữ liệu thu được đi vào từ bên phải, thì các giá trị 1 được dịch sang bên trái. Khi bit khởi đầu đến vị trí trái cùng trong thanh ghi dịch (mà ở các chế độ 2 và 3 nó là thanh ghi 9 bit), thì nó sẽ chỉ thị cho khối điều khiển RX thực hiện phép chuyển dịch cuối cùng rồi nạp SBUF và RB8 và xác lập cờ ngắt RI.

Tín hiệu để nạp SBUF, RB8 và để xác lập RI sẽ được tạo ra khi và chỉ khi các điều kiện sau đây được thoả mãn ở thời điểm xung nhịp cuối cùng được tạo ra:

1. R1 = 0 và
2. Hoặc SM2 = 0 hoặc bit STOP nhận được = 1.

Nếu một trong những điều kiện này không được thoả mãn, thì khung tin nhận được sẽ bị mất mà không thể phát hiện được và RI cũng không được xác lập. Nếu cả 2 điều kiện được thoả mãn, thì bit dữ liệu thứ 9 nhận được sẽ chuyển vào RB8, còn 8 bit dữ liệu được chuyển vào SBUF. Nếu cả hai điều kiện được thoả mãn, thì bit dừng chuyển vào RB8, 8 bit dữ liệu chuyển vào SBUF và RI được kích hoạt. Ở thời điểm này, bất kể các điều kiện trên được thoả mãn hay không, thì khối điều khiển vẫn quay trở lại để tiếp tục chức năng phát hiện đột biến mới từ 1 xuống 0 trên tuyến thu tin RxD.

9.6. THANH GHI ĐIỀU KHIỂN NGUỒN PCON CỦA HỆ VI XỬ LÝ ON-CHIP 80C51

PCON là thanh ghi điều khiển nguồn 8 bit (hình 9.28).

Bit PD (POWER DOWN) và bit IDL (IDLE) chỉ được sử dụng trong phiên bản công nghệ CMOS của họ 8051, tức là 80C51. Các bit này đặt hệ vi xử lý on-chip vào các chế độ đặc biệt duy trì nguồn cho tới khi hệ vi xử lý on-chip được đưa vào hoạt động.

Hai bit GF1, GF0 là hai bit được sử dụng vào mục đích chung, người lập trình có thể dùng để cất giữ thông tin.

Bit SMOD dùng điều khiển việc lập tốc độ baud.

MSB (trái) … LSB (phải)

SMOD	-	-	-	GF1	GF0	PD	IDL

Bit	Kí hiệu	Chức năng
7	SMOD	Bit điều khiển tốc độ baud cho UART
6, 5, 4- không dùng		
3	GF1	Dùng cho mục đích chung
2	GF0	Dùng cho mục đích chung
1	PD	Bit mất nguồn
0	IDL	Idle mode bit = 1 cho phép kích hoạt việc kiểm soát mất nguồn trong 80C51

CHƯƠNG 10
TẬP LỆNH CỦA HỆ VI XỬ LÝ ON-CHIP 80C51

Sau khi đã nghiên cứu cấu trúc của hệ vi xử lý On-chip và chức năng các thành phần bên trong nó, để điều khiển được hoạt động của On-chip cần nắm được tập lệnh của hệ vi xử lý On-chip và nguyên tắc thực hiện các lệnh đó.

10.1. NGUYÊN LÝ THỰC HIỆN LỆNH CỦA ON-CHIP 80C51

10.1.1. Cấu trúc lệnh của hệ vi xử lý on-chip 80C51

Hoạt động của hệ vi xử lý on-chip là thực hiện các lệnh theo thứ tự của tập lệnh đã cài đặt trong bộ nhớ của hệ hoặc thực hiện lệnh theo điều khiển của cơ chế ngắt.

Cũng giống như bất kỳ tập lệnh cho các bộ vi xử lý khác, mỗi lệnh của hệ vi xử lý On-chip cũng được xử lý theo chu kỳ lệnh gồm hai giai đoạn là giai đoạn gọi lệnh và giai đoạn thực hiện lệnh. Mỗi lệnh gồm hai phần, phần thứ nhất là mã lệnh (opcode), phần thứ hai là toán hạng (operand). Mã lệnh là phần đầu tiên của lệnh, nó cho biết ý nghĩa của lệnh và độ dài của câu lệnh. Toán hạng là phần thứ hai của lệnh. Độ dài của câu lệnh tuỳ thuộc vào mỗi lệnh. Trong tập lệnh của hệ vi xử lý on-chip 80C51 có lệnh dài 1 byte, có lệnh dài 2 byte và có lệnh dài tới 3 byte. Những lệnh có độ dài 2 hoặc 3 byte thì byte đầu tiên luôn luôn là byte mã lệnh, các byte tiếp theo là các toán hạng. Với lệnh dài 1 byte thì cả mã lệnh và toán hạng cùng nằm trong byte lệnh .

Mỗi chu kỳ lệnh của hệ vi xử lý On-chip phải thực hiện trong nhiều chu kỳ máy, số chu kỳ máy trong một chu kỳ lệnh tuỳ thuộc vào mỗi lệnh. Có lệnh dài 1 byte thì thực hiện trong 1 chu kỳ máy, có lệnh dài 2 hoặc 3 byte thì thực hiện trong 2 chu kỳ máy, lại có lệnh dài 1 byte lại thực hiện trong 2, 3 hoặc đến 4 chu kỳ máy nhưng có lệnh dài 2 hoặc 3 byte thì chỉ thực hiện trong 1 chu kỳ máy.

10.1.2. Xử lý lệnh của hệ vi xử lý on-chip 80C51

Giai đoạn gọi lệnh

Giai đoạn đầu tiên của quá trình xử lý lệnh là giai đoạn gọi lệnh từ bộ nhớ chương trình. Nội dung thanh ghi con trỏ chương trình PC chính là địa chỉ của lệnh cần xử lý được xuất ra theo kênh địa chỉ để xác định ngăn nhớ chứa byte mã lệnh cần đưa vào CPU của hệ. Tuỳ theo bộ nhớ chương trình là nội trú hay ngoại trú mà hệ vi xử lý on-chip thực hiện phương thức truy xuất khác nhau. Nếu truy xuất bộ nhớ chương trình là nội trú thì byte mã lệnh sẽ trực tiếp được xuất ra theo kênh dữ liệu để chuyển tới thanh ghi lệnh. Nếu truy xuất bộ nhớ chương trình là ngoại trú thì địa chỉ của lệnh cần xử lý được chuyển qua cổng P0 và cổng P2 ra ngoài, đồng thời khối điều khiển và đồng bộ phát ra tín hiệu </PSEN> ở ngay chu kỳ đầu gọi lệnh để chọn bộ nhớ chương trình ngoài, tiếp theo tín hiệu <ALE> cũng được phát ra trong thời gian chu kỳ dao động thứ 2 và thứ 3 của chu kỳ máy thứ nhất để chốt byte thấp của địa chỉ. Khi được truy cập, byte mã lệnh từ bộ nhớ chương trình được xuất ra theo kênh dữ liệu, qua cổng P0 và qua chốt cổng 0 để chuyển vào thanh ghi lệnh. Byte mã lệnh từ thanh ghi lệnh được chuyển sang bộ giải mã lệnh để xác định ý nghĩa của lệnh và độ dài của câu lệnh, nhờ đó mà khối điều khiển và đồng bộ sẽ xác định công việc mà vi xử lý on-chip phải thực hiện tiếp theo. Đồng thời với quá trình giải mã lệnh bộ đếm chương trình tự động tăng thêm 1 đơn vị để xác định địa chỉ của lệnh cần xử lý tiếp theo trong <PC> nếu lệnh vừa xử lý dài 1 byte, hoặc xác định địa chỉ của các toán hạng nếu lệnh đang xử lý dài 2 hoặc 3 byte và quá trình gọi các byte tiếp theo của lệnh cũng giống như quá trình gọi byte mã lệnh.

Giai đoạn thực hiện lệnh

Giai đoạn thứ hai của quá trình xử lý một lệnh là giai đoạn thực hiện lệnh. Các lệnh khác nhau sẽ có nguyên lý thực hiện lệnh khác nhau. Phương thức thực hiện lệnh, thời điểm và khoảng thời gian thực hiện lệnh tuỳ thuộc vào ý nghĩa và chức năng của từng lệnh (có lệnh dài 1 byte 1 chu kỳ máy, lệnh dài 1 byte 2 chu kỳ máy và có lệnh dài 2 byte 1 chu kỳ máy, lệnh dài 2 byte 2 chu kỳ máy hoặc dạng lệnh có địa chỉ trực tiếp, địa chỉ gián tiếp, lệnh làm thay đổi nội dung trong các thanh ghi chức năng, lệnh truy cập bộ nhớ dữ liệu).

- *Thực hiện với lệnh dài 1 byte, 1 chu kỳ máy*: Với loại lệnh này, giai đoạn gọi lệnh và giai đoạn thực hiện lệnh đều thực hiện trong một chu kỳ máy. Thời gian gọi lệnh bắt đầu từ đầu trạng thái S1 đến cuối trạng thái S3 và thời gian thực hiện lệnh bắt đầu từ đầu trạng thái S4 đến cuối trạng thái S6 của chu kỳ máy. Hệ vi xử lý On-chip 80C51 có các lệnh thuộc loại này thường là các lệnh trong các nhóm lệnh chuyển dữ

liệu, thực hiện tính toán logic và số học, điều khiển biến mà các toán hạng trong câu lệnh là thanh ghi tích luỹ Acc, các thanh ghi Rn của băng thanh ghi hiện hành hoặc là địa chỉ các ô nhớ của bộ nhớ dữ liệu nội trú được đánh địa chỉ thông qua thanh ghi R0 hoặc R1 (kiểu @Ri).

- *Thực hiện với lệnh dài 1 byte, 2 hoặc 4 chu kỳ máy*: Là các lệnh truy cập bộ nhớ dữ liệu ngoại trú, lệnh tăng con trỏ dữ liệu (DPTR) hay lệnh xử lý dữ liệu 16 bit. Sau thời gian gọi lệnh, từ đầu trạng thái S4 của chu kỳ máy đầu tiên (có một số lệnh phải từ đầu trạng thái S4 của chu kỳ máy thứ hai) cho đến hết chu kỳ máy cuối sẽ diễn ra quá trình thực hiện lệnh. Đối với các lệnh truy cập bộ nhớ dữ liệu ngoại trú, khi thực hiện lệnh thì khối đồng bộ và điều khiển sẽ phát ra tín hiệu chốt địa chỉ <ALE> trong khoảng thời gian từ P2-S4 đến P2-S5 của chu kỳ máy, phát tín hiệu </RD> hoặc </WR> trong khoảng thời gian từ P1-S1 đến P2-S3 của chu kỳ máy thứ hai hay các chu kỳ máy tiếp theo trong chu kỳ lệnh để điều khiển đọc bộ nhớ hoặc ghi vào bộ nhớ dữ liệu.
- *Thực hiện với lệnh dài 2 byte, 1 chu kỳ máy*: Hệ vi xử lý On-chip 80C51 có các lệnh trong các nhóm lệnh chuyển dữ liệu, thực hiện tính toán logic và số học, điều khiển biến mà toán hạng đích là thanh ghi tích luỹ, toán hạng nguồn là địa chỉ ô nhớ dữ liệu 8 bit của RAM nội trú hoặc địa chỉ các thanh ghi 8 bit trong vùng <SFR>. Một số lệnh trong nhóm lệnh trên chỉ có 1 toán hạng mà toán hạng này là địa chỉ ô nhớ dữ liệu 8 bit của RAM nội trú hoặc địa chỉ các thanh ghi 8 bit trong <SFR>. Lệnh có byte đầu là mã lệnh, byte thứ 2 là toán hạng. Khi byte mã lệnh được gọi thì nó được chuyển vào thanh ghi lệnh và bộ đếm chương trình tự động tăng thêm 1 đơn vị để xác định địa chỉ của byte thứ 2. Khi byte thứ 2 được gọi, nó sẽ được chuyển vào thanh ghi cơ sở và từ đây các ô nhớ dữ liệu 8 bit của RAM hoặc địa chỉ các thanh ghi 8 bit trong <SFR> được địa chỉ hoá. Thời gian gọi lệnh bắt đầu từ đầu trạng thái S1 đến cuối trạng thái S3 và thời gian thực hiện lệnh bắt đầu từ đầu trạng thái S4 đến cuối trạng thái S6 của chu kỳ máy.
- *Thực hiện với lệnh dài 2 byte hoặc 3 byte, 2 chu kỳ máy*: Lệnh này thường là các lệnh chuyển dữ liệu giữa các ô nhớ của bộ nhớ nội trú với nhau hoặc chuyển dữ liệu là hằng số có trong câu lệnh vào các ô nhớ của bộ nhớ nội trú hay các lệnh nhảy. Các lệnh này có byte thứ 2 hoặc byte thứ 2 và thứ 3 là các byte toán hạng sẽ được thực hiện ở cuối chu kỳ máy thứ nhất và cả chu kỳ máy thứ 2.

10.2. TỔ CHỨC KHÔNG GIAN BỘ NHỚ CỦA ON-CHIP 80C51

Để có thể hiểu các lệnh của hệ vi xử lý On-chip hoạt động như thế nào trong môi trường của hệ vi xử lý cần biết cách tổ chức vùng nhớ của hệ, vì nó là

thành phần không thể thiếu của các quá trình xử lý thông tin xảy ra trong hệ vi xử lý On-chip.

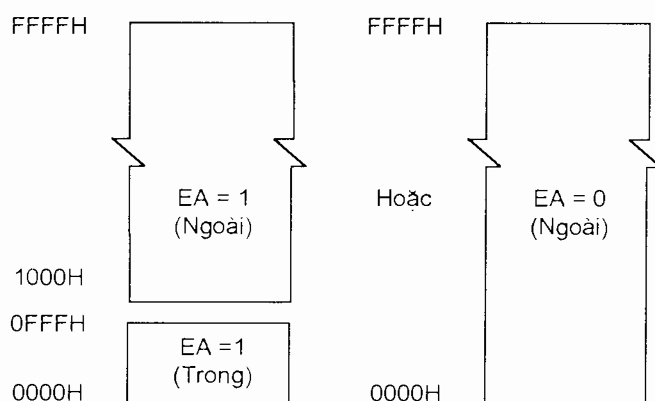
Về cấu trúc, Hệ vi xử lý On-chip có vùng không gian bộ nhớ được định địa chỉ riêng biệt cho bộ nhớ chương trình <ROM> và bộ nhớ dữ liệu <RAM>.

10.2.1. Bộ nhớ chương trình EPROM

+ Bộ nhớ chương trình <EPROM> có vùng không gian nhớ như biểu diễn trên hình 10.1. Không gian cực đại của bộ nhớ này chiếm tới 64kb, được định địa chỉ từ 0000h đến FFFFh trong đó có 4kb nội trú bên trong hệ vi xử lý on-chip được định địa chỉ từ 0000h đến 0FFFh và mở rộng thêm 60kb bộ nhớ chương trình bên ngoài được định địa chỉ từ 1000h đến FFFFh.

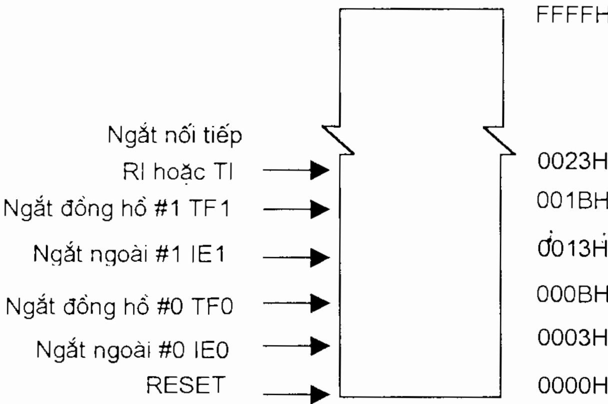
+ On-chip 80C51 cũng có thể sử dụng toàn bộ bộ nhớ chương trình ngoài gồm 64kb được định địa chỉ từ 0000h đến FFFFh.

+ Việc lựa chọn bộ nhớ chương trình nội trú, bộ nhớ chương trình mở rộng ngoại trú hoặc toàn bộ bộ nhớ chương trình ngoại trú bên ngoài on-chip được thực hiện bằng tín hiệu chọn cách truy xuất /EA (*external access*). Khi chân /EA của on-chip được gán mức logic 1 thì hệ vi xử lý on-chip sử dụng vừa bộ nhớ chương trình nội trú vừa bộ nhớ chương trình ngoại trú. Khi chân /EA định vị ở mức logic thấp thì hệ vi xử lý on-chip chỉ sử dụng bộ nhớ ngoại trú.



Hình 10.1. **Tổ chức không gian của bộ nhớ chương trình**

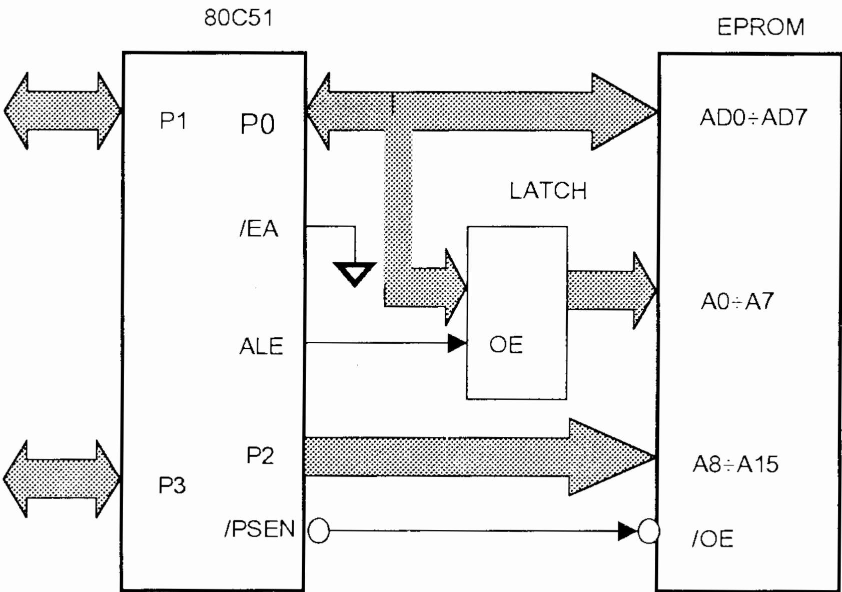
+ Mỗi khi được RESET, hệ vi xử lý on-chip sẽ truy cập bộ nhớ chương trình từ địa chỉ khởi đầu là 0000h, sau đó nếu cơ chế ngắt được sử dụng thì on-chip truy cập tới địa chỉ quy định trong bảng vector ngắt. Các địa chỉ cố định do ngắt định vị như hình 10.2.



Hình 10.2. **bảng vector ngắt của ON-CHIP 80C51**

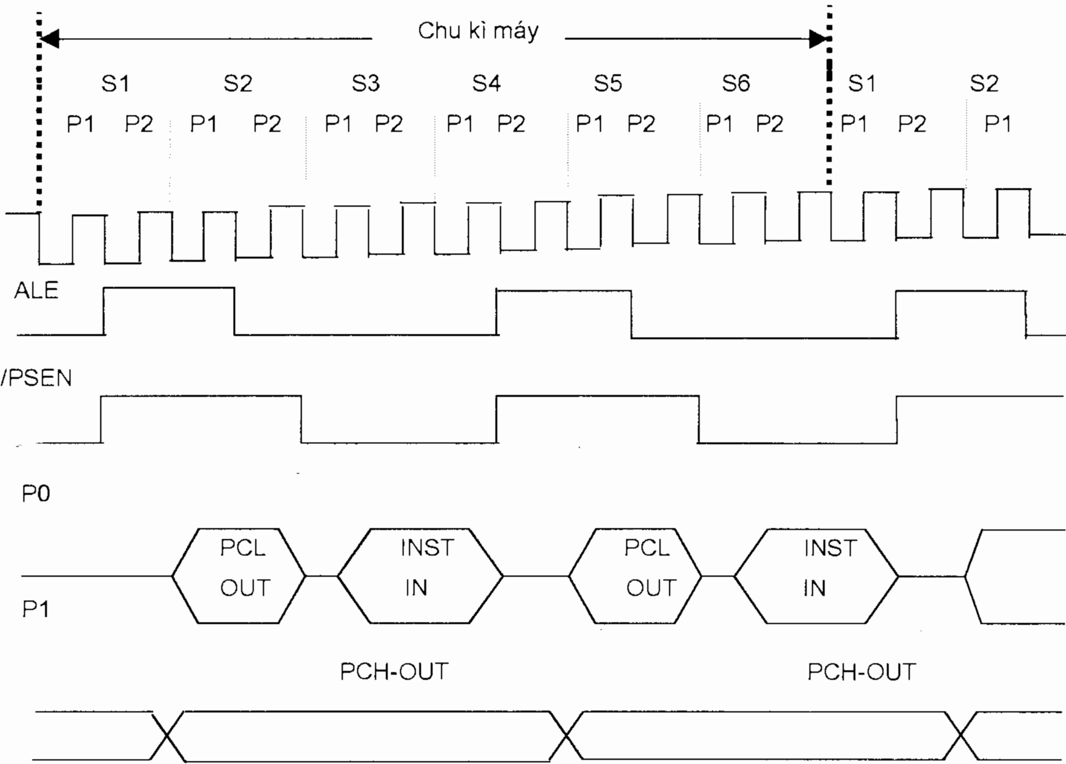
+ Khi truy cập bộ nhớ chương trình, hệ vi xử lý on-chip sử dụng xung chọn /PSEN (Program Strobe Enable) tương đương như xung đọc RD để đọc lệnh. Nếu hệ vi xử lý on-chip làm việc với bộ nhớ chương trình nội trú thì chân phát ra xung chọn /PSEN không sử dụng, còn nếu on-chip làm việc với bộ nhớ chương trình ngoại trú thì chân phát ra xung chọn /PSEN được sử dụng. Khi đó nếu /PSEN = 0 thì cho phép hệ vi xử lý on-chip đọc bộ nhớ chương trình còn khi /PSEN =1 thì không cho phép hệ vi xử lý on-chip chọn bộ nhớ chương trình.

+ Khi hệ vi xử lý On-chip truy cập bộ nhớ chương trình ngoại trú, nó luôn sử dụng địa chỉ 16 bit thông qua cổng P0 và cổng P2 bất kể bộ nhớ chương trình ngoài có dung lượng nhớ là 60kb hay 64kb. Sơ đồ mạch của on-chip khi truy cập bộ nhớ ngoài được thể hiện như hình 10.3.



Hình 10.3. **Sơ đồ tổ chức bộ nhớ chương trình ngoại trú**

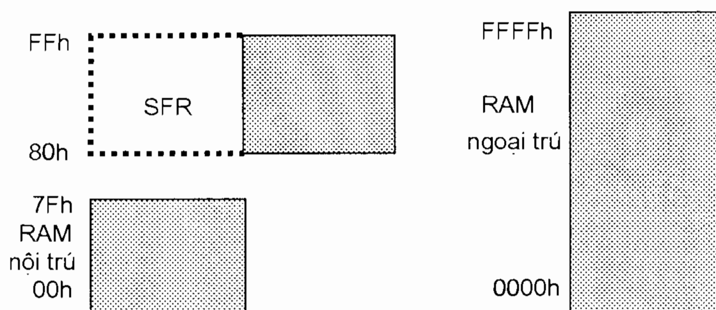
Nguyên lý nhận lệnh từ bộ nhớ chương trình ngoại trú được biểu diễn bằng đồ thị thời gian như được thể hiện trên hình 10.4. Khi truy cập bộ nhớ chương trình ngoại trú, hệ vi xử lý on-chip phát xung chốt địa chỉ <ALE>. Mỗi chu kỳ máy gồm 2 xung chốt, mỗi xung chốt tồn tại trong 2 chu kỳ dao động từ P2-S1 đến P1-S2 và từ P2-S4 đến P1-S5. Để địa chỉ hoá bộ nhớ chương trình ngoại trú, byte thấp của địa chỉ từ bộ đếm chương trình <PCL-OUT> của on-chip được xuất ra qua cổng P0 tại các trạng thái S2 và S5 của chu kỳ máy, byte cao của địa chỉ từ bộ đếm chương trình <PCH-OUT> của on-chip được xuất ra qua cổng P2 trong khoảng thời gian của cả chu kỳ máy. Tiếp theo xung chốt, hệ vi xử lý On-chip 80C51 phát ra xung đọc /PSEN. Mỗi chu kỳ máy của chu kỳ lệnh gồm 2 xung đọc /PSEN, mỗi xung đọc /PSEN tồn tại trong 3 chu kỳ dao động từ P1-S3 đến hết P1-S4 và từ P1-S6 đến hết P1-S1 của chu kỳ máy tiếp theo. Trong khoảng thời gian phát xung đọc /PSEN thì byte mã lệnh <INST-IN> được đọc từ bộ nhớ chương trình để nhập vào on-chip.



Hình 10.4. **Đồ thị thời gian nhận lệnh từ bộ nhớ chương trình ngoại trú**

10.2.2. Bộ nhớ dữ liệu RAM

Hệ vi xử lý On-chip 80C51 có bộ nhớ dữ liệu RAM chiếm một không gian bộ nhớ độc lập với bộ nhớ chương trình EPROM. Mô tả không gian bộ nhớ dữ liệu của hệ vi xử lý On-chip như thể hiện trên hình 10.5.



Hình 10.5. **Không gian của bộ nhớ dữ liệu RAM**

+ Trong on-chip chứa bộ nhớ dữ liệu RAM có dung lượng là 128 byte được định vị địa chỉ từ 00h đến 7Fh và có một vùng không gian bộ nhớ được định vị địa chỉ từ 80h đến FFh dành riêng cho các thanh ghi chức năng đặc biệt <SFR>. Bộ nhớ dữ liệu nội trú của on-chip còn có thể có vùng không gian nhớ với dung lượng 128 byte được định vị địa chỉ từ 80h đến FFh song song với vùng <SFR>.

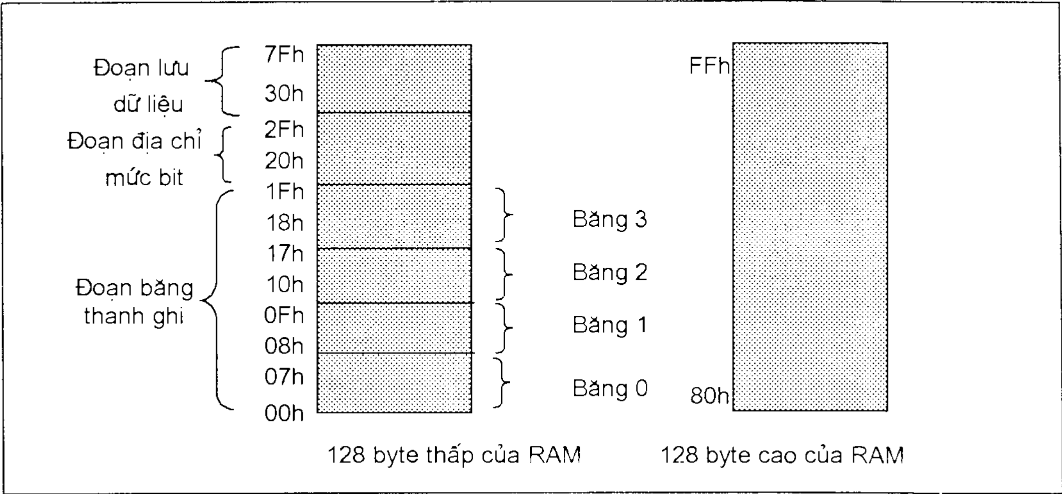
+ On-chip 80C51 cũng có thể làm việc với bộ nhớ RAM ngoại trú dung lượng cực đại 64kb được định địa chỉ từ 1000h đến FFFFh.

Bộ nhớ dữ liệu nội trú

Bộ nhớ dữ liệu có khoảng không gian nhớ gồm 128byte được định địa chỉ từ 00h đến 7Fh được chia thành 3 đoạn.

Đoạn thứ nhất gồm 32 byte có địa chỉ từ 00h đến 1Fh lại được chia thành 4 băng thanh ghi, mỗi băng gồm 8 thanh ghi 8 bit, địa chỉ của mỗi băng thanh ghi được xác định như hình 10.6. Các thanh ghi trong mỗi băng có tên gọi từ R0 cho đến R7. RAM gồm 128 byte thấp này đều được truy cập bằng địa chỉ trực tiếp mức byte. Băng thanh ghi được quy định bởi 2 bit RS0 và RS1 của thanh ghi từ trạng thái chương trình <PSW>.

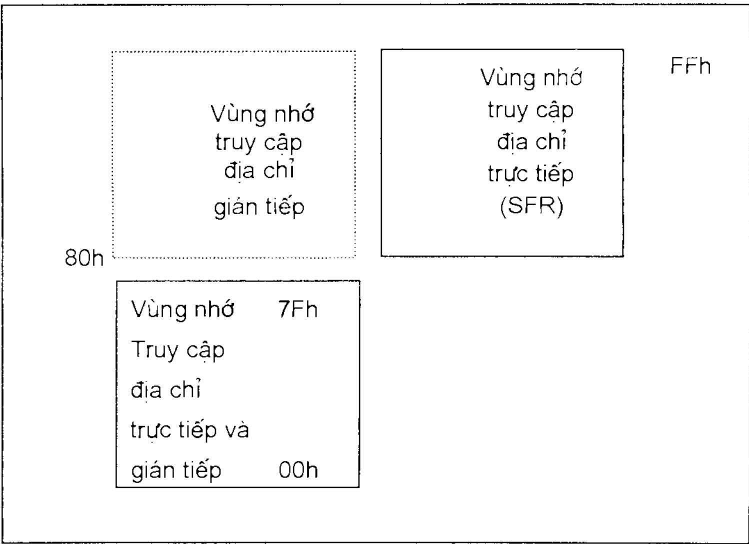
Đoạn thứ hai gồm 16 byte có địa chỉ từ 20h đến 2Fh được phép truy cập bằng địa chỉ trực tiếp mức bit. Hệ vi xử lý On-chip có các lệnh được sử dụng để truy cập tới 128 bit trong đoạn bộ nhớ này sẽ chứa địa chỉ mức bit và được đánh địa chỉ từ 00h đến 7Fh.



Hình 10.6. **Phân đoạn bộ nhớ RAM trong on-chip**

Đoạn thứ ba gồm 80 byte có địa chỉ từ 20h đến 7Fh được dành riêng cho người sử dụng để lưu trữ dữ liệu. Đoạn bộ nhớ này được truy cập bằng địa chỉ mức byte dạng trực tiếp hoặc gián tiếp thông qua các băng thanh ghi.

Bộ nhớ dữ liệu có không gian nhớ phía trên gồm 128byte được định địa chỉ từ 80h đến FFh. Tuỳ theo từng loại on-chip trong họ hệ vi xử lý on-chip 8051 mà chúng có thể được tổ chức thành 2 vùng nhớ có địa chỉ song song. Nguyên tắc truy cập địa chỉ của từng vùng nhớ này khác nhau. Một vùng nhớ được truy cập bằng địa chỉ trực tiếp, một vùng nhớ được truy cập bằng địa chỉ gián tiếp.



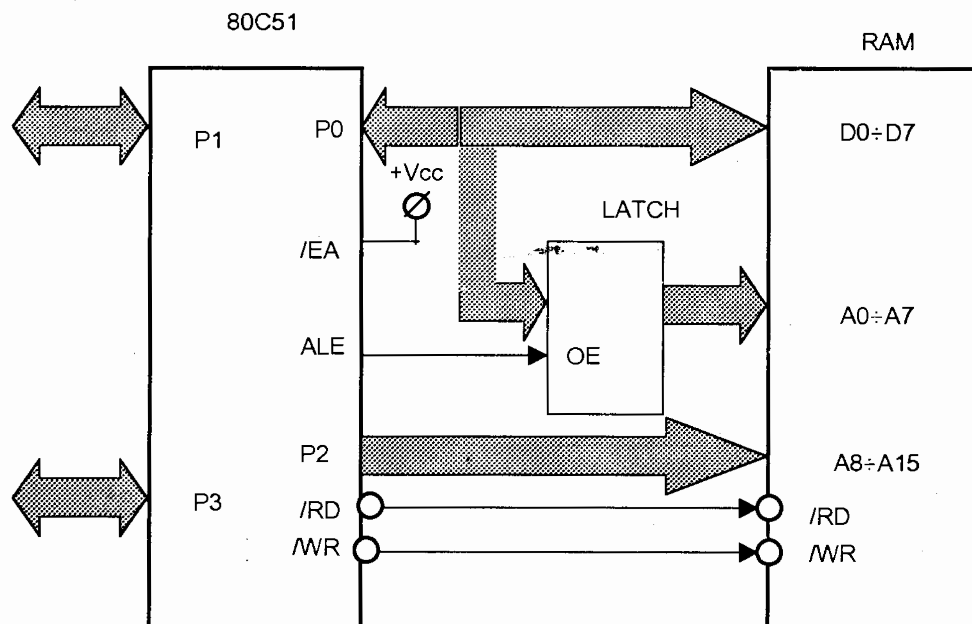
Hình 10.7. **Vùng nhớ dữ liệu 128 byte trên của RAM nội trú**

Vùng nhớ được truy cập bằng địa chỉ trực tiếp là các thanh ghi chức năng đặc biệt <SFR>.

Vùng nhớ được truy cập bằng địa chỉ gián tiếp dành riêng cho lĩnh vực ứng dụng lưu trữ dữ liệu.

Bộ nhớ dữ liệu ngoại trú

Hệ vi xử lý On-chip truy cập bộ nhớ dữ liệu ngoại trú bằng địa chỉ có độ dài 2 byte hoặc 1 byte. Bộ nhớ có địa chỉ dài 2 byte được on-chip truy cập thông qua cổng P0 và cổng P2.Trường hợp bộ nhớ ngoài có dung lượng chỉ cần truy cập bằng địa chỉ dài 1 byte thì on-chip cấp địa chỉ thông qua cổng 0. Sơ đồ mạch của on-chip truy nhập bộ dữ liệu ngoài như thể hiện trên hình 10.8. Sơ đồ mạch này dùng với bộ nhớ chương trình nội trú trong hệ vi xử lý on-chip cho nên chân </EA> được nối vào +Vcc.

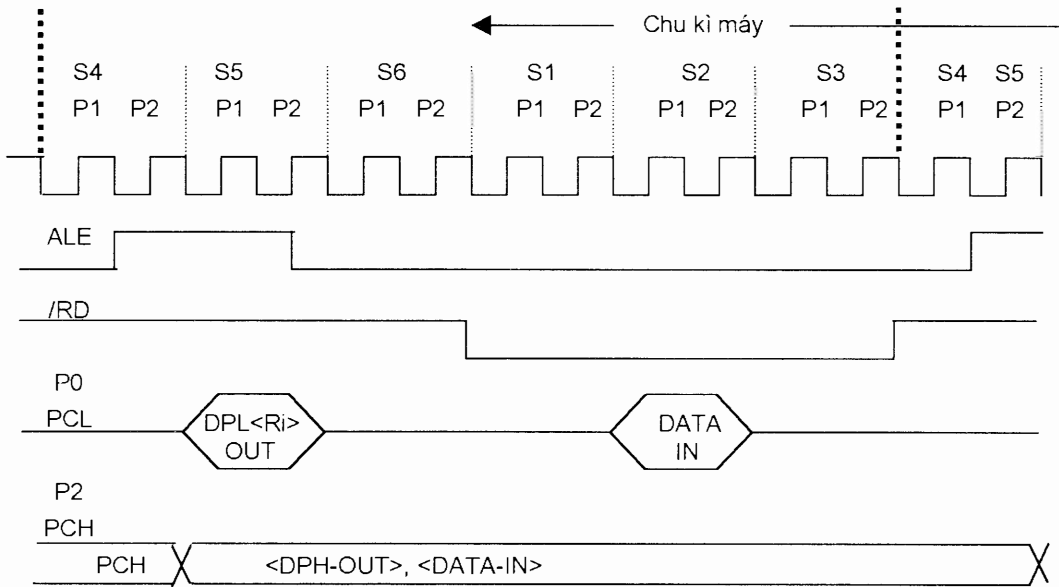


Hình 10.8. **Sơ đồ tổ chức bộ nhớ dữ liệu ngoại trú**

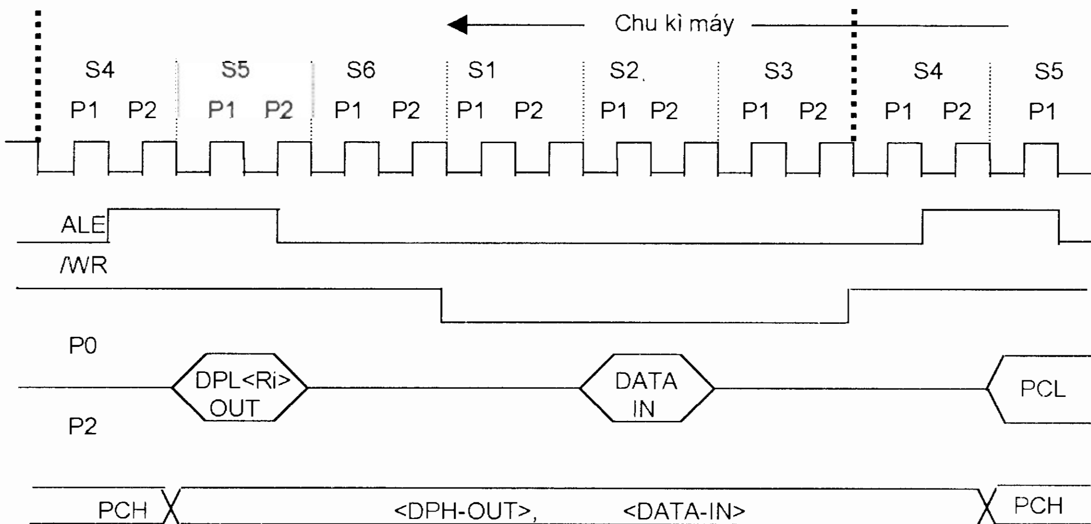
Nguyên lý truy cập bộ nhớ dữ liệu ngoại trú được biểu diễn bằng đồ thị thời gian theo hình 10.9 và 10.10. Khi truy cập bộ nhớ dữ liệu ngoại trú, tuỳ thuộc vào nhiệm vụ đọc dữ liệu từ bộ nhớ hay ghi dữ liệu vào bộ nhớ mà nguyên lý truy cập bộ nhớ dữ liệu có khác nhau.

Quá trình đọc dữ liệu từ bộ nhớ ngoài on-chip phải phát xung chốt địa chỉ <ALE> cho bộ chốt bên ngoài. Mỗi chu kỳ máy có 1 xung chốt tồn tại trong 2 chu kỳ dao động từ P2-S4 đến P1-S5. Để địa chỉ hoá bộ nhớ dữ liệu ngoài, byte thấp của địa chỉ từ con trỏ dữ liệu <DPL> của on-chip được xuất ra qua cổng P0 trong khoảng các trạng thái S5 của chu kỳ máy trong chu kỳ lệnh, tiếp theo cũng ở cổng P0 byte thấp của địa chỉ từ <PCL> xuất ra truy cập bộ nhớ chương trình để thực hiện lệnh tiếp theo. Byte cao của địa chỉ từ con trỏ dữ liệu <DPH> của on-chip được xuất ra qua cổng P2 trong khoảng thời gian từ S5 đến S4 của chu kỳ máy tiếp theo. Ngoài khoảng thời gian này cổng P2 xuất địa chỉ từ <PCH> đến bộ nhớ chương trình. Trường hợp địa chỉ hoá bộ nhớ có địa chỉ dài 1 byte thì địa

chỉ có thể từ con trỏ dữ liệu <DPL> hoặc từ thanh ghi nào đó trong băng thanh ghi được chọn xuất ra qua cổng P0. Tiếp theo xung chốt, On-chip 80C51 phát ra tín hiệu điều khiển </RD> cho phép đọc dữ liệu từ bộ nhớ. Xung </RD> tồn tại trong 3 trạng thái của mỗi chu kỳ máy từ P1-S1 đến P2-S3 và trong khoảng thời gian này dữ liệu từ bộ nhớ được đọc vào on-chip.



Hình 10.9. **Đồ thị thời gian chu kỳ đọc dữ liệu từ bộ nhớ ngoài**



Hình 10.10. **Đồ thị thời gian chu kỳ ghi dữ liệu vào bộ nhớ ngoài**

Quá trình ghi dữ liệu vào bộ nhớ ngoại trú có nguyên lý thể hiện trên đồ thị thời gian hình 10.11. Tương tự như quá trình đọc dữ liệu từ bộ nhớ ngoại trú, chỉ khác là thay vì tín hiệu điều khiển đọc </RD> bằng tín hiệu điều khiển ghi </WR>.

Bộ nhớ dữ liệu nội trú bên trong và ngoại trú bên ngoài của on-chip 80C51 tuy có vùng không gian được tổ chức song song nhưng việc truy cập tới các bộ nhớ này hoàn toàn tách biệt được thực hiện bởi những lệnh khác nhau.

Những lệnh sử dụng để truy cập bộ nhớ dữ liệu nội trú bên trong bao gồm:

MOV A,<src> và MOV <dest>,<src>: Lệnh chuyển dữ liệu từ toán hạng nguồn vào toán hạng đích là thanh ghi tích luỹ, là các ô nhớ hoặc thanh ghi trong on-chip có địa chỉ trực tiếp hoặc gián tiếp. Toán hạng nguồn có thể là các ô nhớ hoặc thanh ghi trong on-chip có địa chỉ trực tiếp hoặc gián tiếp, có thể là giá trị trực hằng chứa trong câu lệnh.

MOV <dest>,A: Lệnh chuyển dữ liệu từ toán hạng nguồn là thanh ghi tích luỹ vào toán hạng đích là các ô nhớ, là các thanh ghi trong on-chip có địa chỉ trực tiếp hoặc gián tiếp.

MOV DPTR,#data16: Lệnh chuyển dữ liệu từ toán hạng nguồn là giá trị hằng dài 16 bit chứa trong câu lệnh vào toán hạng đích là thanh ghi con trỏ dữ liệu.

PUSH <src>: Lệnh chuyển giá trị từ toán hạng nguồn vào ngăn xếp với toán hạng nguồn là ô nhớ hay thanh ghi có địa chỉ trực tiếp.

POP <dest>: Lệnh chuyển giá trị từ ngăn xếp vào toán hạng đích với toán hạng đích là ô nhớ hay thanh ghi có địa chỉ trực tiếp.

XCH A,<byte>: Lệnh chuyển đổi dữ liệu giữa toán hạng nguồn là các ô nhớ, thanh ghi có địa chỉ trực tiếp hoặc gián tiếp với thanh ghi tích luỹ.

XCHD A,@Ri: Lệnh chuyển đổi nửa thấp của thanh ghi tích luỹ với toán hạng nguồn là các ô nhớ, là các thanh ghi có địa chỉ gián tiếp.

Những lệnh sử dụng để truy cập bộ nhớ dữ liệu ngoại trú bao gồm :

MOVX A,@Ri: Lệnh chuyển dữ liệu 8 bit từ ô nhớ của RAM ngoài có địa chỉ xác định trong thanh ghi của băng thanh ghi hiện hành vào thanh ghi tích luỹ.

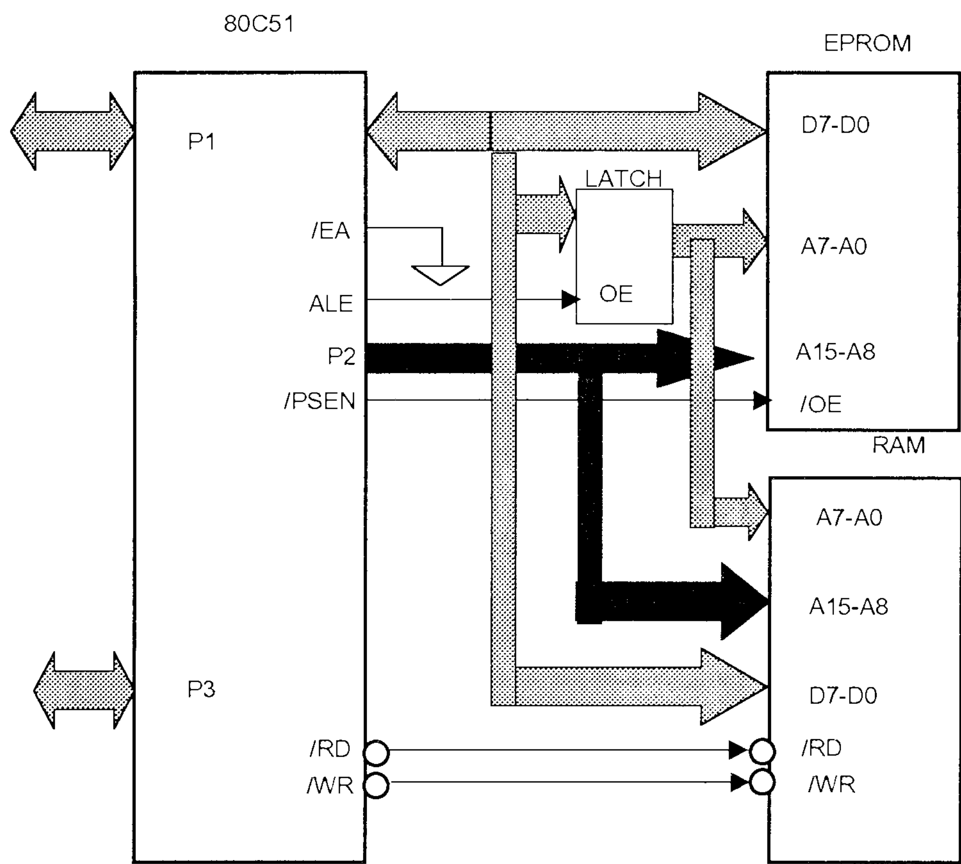
MOVX @Ri,A: Lệnh chuyển dữ liệu 8 bit từ thanh ghi tích luỹ vào ô nhớ của RAM ngoài có địa chỉ xác định trong thanh ghi của băng thanh ghi hiện hành.

MOVX A,@DPTR: Lệnh chuyển dữ liệu 16 bit từ ô nhớ của RAM ngoài có địa chỉ xác định trong thanh ghi con trỏ dữ liệu vào thanh ghi tích luỹ.

MOVX @DPTR,A: Lệnh chuyển dữ liệu 16 bit từ thanh ghi tích luỹ vào ô nhớ của RAM ngoài có địa chỉ xác định trong thanh ghi con trỏ dữ liệu.

Ngoài phương pháp dùng bộ nhớ chương trình nội trú và bộ nhớ dữ liệu ngoại trú hoặc ngược lại hay sử dụng cả bộ nhớ chương trình và bộ nhớ dữ liệu

cùng nội trú, on-chip còn có thể sử dụng toàn bộ bộ nhớ bên ngoài có sơ đồ mạch như hình 10.11.



Hình 10.11. **Sơ đồ tổ chức bộ nhớ ngoại trú**

10.3. TẬP LỆNH CỦA HỆ VI XỬ LÝ ON-CHIP 80C51

Toàn bộ tập lệnh của hệ vi xử lý on-chip 80C51 được trình bày chi tiết trong mục này.

Một số quy định trong câu lệnh và địa chỉ

+ Rn: Thanh ghi R0-R7 của băng thanh ghi hiện hành đang được chọn.
+ Direct: Địa chỉ 8 bit của ô nhớ dữ liệu nội trú, nó có thể là ô nhớ dữ liệu <RAM> hay các thanh ghi chức năng đặc biệt.
+ Ri: Ô nhớ 8bit của bộ nhớ dữ liệu nội trú được định địa chỉ thông qua thanh ghi R1 hoặc R0.
+ #Data: Hằng số 8 bit được chứa trong câu lệnh.
+ #Data: Hằng số 16 bit được chứa trong câu lệnh.

+ Addr16 : Địa chỉ đích 16 bit của 64kb không gian bộ nhớ chương trình, được lệnh LCALL và lệnh LJMP sử dụng.
+ Addr11: Địa chỉ đích 11bit của 2kb của bộ nhớ chương trình, được lệnh ACALL và lệnh AMP sử dụng.
+ RELL: Đánh dấu byte offset 8bit (bù 2), nó được lệnh SJMP và tất cả các lệnh nhảy có điều kiện sử dụng. Giá trị của nó từ -128 đến + 127.
+ Bit: Bit được định địa chỉ trực tiếp trong RAM nội trú hoặc các thanh ghi chức năng đặc biệt.

10.3.1. Nhóm lệnh chuyển dữ liệu

Lệnh MOV dạng byte

Cú pháp câu lệnh: **MOV <dest-byte >, <src-byte>**

Chức năng: Lệnh sao chép nội dung của toán hạng nguồn vào toán hạng đích, nội dung của toán hạng nguồn không thay đổi. Lệnh này không làm ảnh hưởng tới các cờ.

+ MOV A, Rn : (A) ← (Rn)

  Kích thước lệnh: 1 byte.

  Thời gian thực hiện: 1 chu kỳ máy.

  Mã lệnh : 1 1 1 0 1 r r r

+ MOV A, direct : (A) ← (direct).

  Kích thước lệnh : Dài 2 byte

  Thời gian thực hiện : 1 chu kỳ máy .

  Mã lệnh :E5h

+ MOV A, @Ri (A) ← ((Ri))

  Kích thước lệnh : Dài 1 byte

  Thời gian thực hiện : 1 chu kỳ máy .

  Mã lệnh : E7h

+ MOV A,#direct : (A) ← (direct).

  Kích thước lệnh : Dài 2 byte.

  Thời gian thực hiện : 1 chu kỳ máy .

  Mã lệnh : 74h

+ MOV Rn, A : (Rn) ← (A)

  Kích thước lệnh : Dài 1 byte.

  Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 1 1 1 1 1 r r r

+ MOV Rn, direct : (Rn) ← (direct).

Kích thước lệnh : Dài 2 byte

Thời gian thực hiện : 2 chu kỳ máy.

Mã lệnh : 1 0 1 0 1 r r r

+ MOV Rn ,#data : (Rn) ← (data)

Kích thước lệnh : Dài 2 byte.

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 01 1 1 1 r r r

+ MOV direct,Rn: (direct) ← (Rn).

Kích thước lệnh : Dài 2 byte

Thời gian thực hiện : 2 chu kỳ máy .

Mã lệnh :1 0 0 0 1 r r r

+ MOV direct,direct : (direct) ← (direct).

Kích thước lệnh : Dài 3 byte

Thời gian thực hiện : 2 chu kỳ máy .

Mã lệnh : 85h

+ MOV direct,@Ri : (direct) ← ((Ri)).

Kích thước lệnh : Dài 2 byte

Thời gian thực hiện : 2 chu kỳ máy .

Mã lệnh : 87h

+ MOV direct,#data : (direct) ← (#data).

Kích thước lệnh : Dài 3 byte

Thời gian thực hiện : 2 chu kỳ máy.

Mã lệnh : 75h

+ MOV @Ri,A : ((Ri)) ← (A)

Kích thước lệnh : Dài 1 byte

Thời gian thực hiện : 1 chu kỳ máy.

Mã lệnh : F7h

+ MOV @Ri, direct : ((Ri)) ← (direct)

Kích thước lệnh : Dài 2byte.

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh : A7h

+ MOV @Ri, #data : ((Ri)) ← #data

Kích thước lệnh : Dài 2byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 77h

Lệnh MOV dạng bit

Cú pháp câu lệnh: MOV <dest-bit >,<source- bit>

Chức năng: Chuyển bit dữ liệu ở dạng sao chép toán hạng nguồn vào toán hạng đích, một toán hạng là cờ hiệu và một toán hạng sẽ là bít địa chỉ trực tiếp. Lệnh này không làm ảnh hưởng tới thanh ghi hoặc các cờ khác.

+ MOV C, bit : (C) ← (bit)

Kích thước lệnh : Dài 2 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 1 0 1 0 0 0 1 0 bit địa chỉ

+ MOV bit, C ; (bit) ← (C)

Kích thước lệnh : Dài 2 byte

Thời gian thực hiện : 2 chu kỳ máy.

Mã lệnh : 1 0 0 1 0 0 1 0 bit địa chỉ

Lệnh MOV dạng Word

Cú pháp câu lệnh: MOV DPTR, #data16 : (DPTR) ← (#data)

Chức năng: Giá trị 16 bit chính là toán hạng thứ hai trực tiếp trong câu lệnh được nạp vào thanh ghi con trỏ dữ liệu <DPTR>, byte cao của dữ liệu được nạp vào thanh ghi <DPH>, còn byte thấp được nạp vào thanh ghi <DPL>. Lệnh này không ảnh hưởng tới các cờ hiệu.

Kích thước lệnh : Dài 3byte

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh : 1001 0000 byte cao byte thấp

Lệnh chuyển byte mã lệnh

Cú pháp câu lệnh: MOVC A , @A + <thanh ghi cơ sở >

Chức năng: Nạp cho thanh ghi tích lũy byte mã lệnh từ bộ nhớ chương trình mà địa chỉ của byte mã lệnh trong bộ nhớ là tổng nội dung của thanh ghi tích lũy gồm 8bit với nội dung của thanh ghi cơ sở hoặc thanh đếm chương trình gồm 16bit.

+ MOVC A,@A+DPTR : (A) ← (A)+(DPTR).

Kích thước lệnh : Dài 1byte

Thời gian thực hiện : 2 chu kỳ máy .

Mã lệnh : 1 0 0 1 0 0 1 1

+ MOVC A,@A+ PC ; (PC) ← (PC) + 1

; (PC) ← ((A)+ (PC))

Kích thước lệnh : Dài 1byte

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh : 10 0 0 0 0 0 1 1

Lệnh chuyển dữ liệu ra ngoài.

Cú pháp câu lệnh: MOVX <dest-byte >,<source - byte >

Chức năng: Lệnh MOVX chuyển dữ liệu giữa thanh ghi tích luỹ với ô nhớ của RAM ngoại trú .

- Nếu dữ liệu được chuyển là 8 bit thì nội dung của R0 hoặc R1 trong băng thanh ghi hiện hành cho địa chỉ 8 bit được đưa ra cổng P0.

– Nếu dữ liệu được chuyển là 16 bit thì con trỏ dữ liệu phát ra địa chỉ 16 bit ,8 bit cao của địa chỉ được chuyển ra chốt cổng P2, còn 8 bit thấp của địa chỉ được phân kênh với 8 bit dữ liệu ở chốt cổng P0.

+ MOVX A,@Ri ; (A) ← ((Ri))

Kích thước lệnh: Dài 1byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 1 1 0 0 0 1 1

+ MOVX A,@DPTR ; (A) ← ((DPTR))

Kích thước lệnh: Dài 1byte

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh: 1 1 1 0 0 0 0 0

+ MOVX @Ri, A ; ((Ri)) ← (A)

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh: 1 1 1 1 0 0 1 1

+ MOVX @DPTR,A ; ((DPTR)) ← (A)

Kích thước lệnh: dài 1byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 1 1 1 0 0 0 0

Lệnh chuyển số liệu vào ngăn xếp

Cú pháp câu lệnh: PUSH direct

Chức năng: Để chuyển số liệu có trong câu lệnh vào ngăn xếp, trước tiên con trỏ ngăn xếp SP được tăng lên 1, sau đó số liệu sẽ được chuyển vào đỉnh của ngăn xếp mà địa chỉ đỉnh này được định bởi SP. Ngăn xếp nằm ở RAM trong On chip.

\+ PUSH direct ; (SP) ← (SP)+1

; ((SP)) ← (direct)

Kích thước lệnh : dài 2 byte

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh : 1 1 0 0 0 0 0 0 địa chỉ trực tiếp

Lệnh chuyển số liệu ra khỏi ngăn xếp

Cú pháp câu lệnh: POP direct

Chức năng: Chuyển nội dung của ô nhớ RAM có địa chỉ được con trỏ ngăn xếp trỏ tới đến nơi có địa chỉ trực tiếp trong câu lệnh, sau đó con trỏ ngăn xếp giảm đi 1.

\+ POP direct ; (direct) ← ((SP))

; (SP) ← (SP)-1

Kích thước lệnh: dài 2 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh : 1 1 0 1 0 0 0 0 địa chỉ trực tiếp

Hoán chuyển dữ liệu

Cú pháp câu lệnh: XCH A,<byte>.

Chức năng: Hoán chuyển nội dung giữa thanh ghi tích luỹ với thanh ghi hoặc bộ nhớ có địa chỉ chứa trong toán hạng thứ 2 của câu lệnh, toán hạng thứ 2 có thể là thanh ghi, thanh ghi trực tiếp hoặc thanh ghi gián tiếp.

\+ XCH A,Rn ; (A) ←→ (Rn) .

Kích thước lệnh: Dài 1 byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 0 1 r r r

\+ XCH A,direct ; (A) ←→ (direct) .

Kích thước lệnh: Dài 2byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 0 0 1 0 1 địa chỉ trực tiếp

Lệnh chuyển số liệu vào ngăn xếp

Cú pháp câu lệnh: PUSH direct

Chức năng: Để chuyển số liệu có trong câu lệnh vào ngăn xếp, trước tiên con trỏ ngăn xếp SP được tăng lên 1, sau đó số liệu sẽ được chuyển vào đỉnh của ngăn xếp mà địa chỉ đỉnh này được định bởi SP. Ngăn xếp nằm ở RAM trong On chip.

+ PUSH direct ; (SP) ← (SP)+1

; ((SP)) ← (direct)

Kích thước lệnh : dài 2 byte

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh : 1 1 0 0 0 0 0 0 địa chỉ trực tiếp

Lệnh chuyển số liệu ra khỏi ngăn xếp

Cú pháp câu lệnh: POP direct

Chức năng: Chuyển nội dung của ô nhớ RAM có địa chỉ được con trỏ ngăn xếp trỏ tới đến nơi có địa chỉ trực tiếp trong câu lệnh, sau đó con trỏ ngăn xếp giảm đi 1.

+ POP direct ; (direct) ← ((SP))

; (SP) ← (SP)-1

Kích thước lệnh: dài 2 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh : 1 1 0 1 0 0 0 0 địa chỉ trực tiếp

Hoán chuyển dữ liệu

Cú pháp câu lệnh: XCH A,<byte>.

Chức năng: Hoán chuyển nội dung giữa thanh ghi tích luỹ với thanh ghi hoặc bộ nhớ có địa chỉ chứa trong toán hạng thứ 2 của câu lệnh, toán hạng thứ 2 có thể là thanh ghi, thanh ghi trực tiếp hoặc thanh ghi gián tiếp.

+ XCH A,Rn ; (A) ←→ (Rn) .

Kích thước lệnh: Dài 1 byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 0 1 r r r

+ XCH A,direct ; (A) ←→ (direct) .

Kích thước lệnh: Dài 2byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 0 0 1 0 1 địa chỉ trực tiếp

+ XCH A,@Ri ; (A) ←→ ((Ri)) .

Kích thước lệnh: Dài 1 byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 0 0 1 1 1

Lệnh hoán chuyển 4 bits thấp

Cú pháp câu lệnh: XCHD A, @Ri.

Chức năng: Lệnh hoán chuyển 4 bit thấp nội dung trong thanh ghi tích luỹ với ô nhớ của RAM bên trong, có địa chỉ được chỉ định gián tiếp qua thanh ghi của <SFR>. Lệnh này không làm ảnh hưởng tới trạng thái các cờ .

+ XCHD A,@Ri ; (A3-0) ←→ ((Ri3-0)) .

Kích thước lệnh: Dài 1 byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 1 0 1 1 1

10.3.2. Nhóm lệnh điều khiển biến logic

Lệnh xoá bit

Cú pháp câu lệnh: CLR (bit) .

Chức năng: Lệnh xoá một bit về 0 , với bit được định địa chỉ trực tiếp Lệnh này không làm ảnh hưởng tới trạng thái các cờ .

+ CLR C ; (C) ← 0 .

Kích thước lệnh: Dài 1 byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 0 0 0 1 1 .

+ CLR bit ; (bit) ← 0 .

Kích thước lệnh: Dài 2 byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 0 0 0 1 0 địa chỉ bit.

Lệnh xoá thanh ghi tích luỹ

Cú pháp câu lệnh : CLR A .

Chức năng : Lệnh xoá tất cả các bit của thanh ghi tích luỹ về 0. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

+ CLR A ; (A) ← 0.

Kích thước lệnh: Dài 1 byte.

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 1 0 0 1 0 0.

Lệnh thiết lập bit

Cú pháp câu lệnh: SETB <bit>.

Chức năng: Lệnh thiết lập một bit lên 1 với bit được định địa chỉ trực tiếp. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

+ SETB C ; (C) ← 1.

  Kích thước lệnh : Dài 1 byte.

  Thời gian thực hiện : 1 chu kỳ máy

  Mã lệnh : 1 1 0 1 0 0 1 1.

+ SETB bit ; (bit) ← 1.

  Kích thước lệnh: Dài 2 byte.

  Thời gian thực hiện: 1 chu kỳ máy

  Mã lệnh : 1 1 0 1 0 0 1 0 địa chỉ bit.

Lệnh lấy bù của bit

Cú pháp câu lệnh: CPL <bit>.

Chức năng : Lệnh lấy bù của bit, làm bit đã cho sẽ được lấy bù với bit được định địa chỉ trực tiếp.

+ CPL C ; (C) ← (/C).

  Kích thước lệnh: Dài 1 byte.

  Thời gian thực hiện: 1 chu kỳ máy

  Mã lệnh: 1 0 1 1 0 0 1 1.

+ CPL bit ; (bit) ← (/bit).

  Kích thước lệnh: Dài 2 byte.

  Thời gian thực hiện: 1 chu kỳ máy

  Mã lệnh: 1 0 1 1 0 0 1 0 địa chỉ bit.

Lệnh lấy bù của thanh ghi tích luỹ

Cú pháp câu lệnh: CPL <A>.

Chức năng: Lệnh lấy bù tất cả các bit của thanh ghi tích luỹ. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

+ CPL A ; (A) ← (/A).

Kích thước lệnh : Dài 1 byte.

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 1 1 1 1 0 1 0 0.

10.3.3. Nhóm lệnh rẽ nhánh chương trình

Lệnh gọi tuyệt đối

Cú pháp câu lệnh: ACALL addr11.

Chức năng: Lệnh gọi chương trình con có địa chỉ cho trước. Lệnh này tăng bộ đếm chương trình <PC> thêm 2 đơn vị để giữ địa chỉ của lệnh tiếp theo, sau đó chuyển nội dung 16 bit vào ngăn xếp và tăng con trỏ ngăn xếp lên 2 đơn vị. Địa chỉ đích sẽ được hình thành bằng cách ghép 5 bit cao của <PC> với 3 bit cao của byte mã lệnh và byte thứ 2 của lệnh. Do đó chương trình con phải nằm trong đoạn 2Kb của bộ nhớ chương trình ít phải chứa lệnh đầu tiên của chương trình con này. Lệnh này không làm ảnh hưởng tới trạng thái các cờ .

+ ACALL addr11 (PC) ← (PC)+2
(SP) ← (SP)+1
(SP) ← (PC7-0)
(SP) ← (SP)+1
(SP) ← (PC15-8)
(PC10-0) ← (page address)

Kích thước lệnh: Dài 2 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: a10 a9 a8 1 0 0 1 1 a7 a6 a5 a4 a3 a2 a1 a0

Lệnh gọi dài

Cú pháp câu lệnh: LCALL addr16.

Chức năng: Lệnh gọi chương trình con có địa chỉ cho trước. Lệnh này tăng bộ đếm chương trình <PC> thêm 3 đơn vị để giữ địa chỉ của lệnh tiếp theo, sau đó chuyển nội dung 16 bit vào ngăn xếp và tăng con trỏ ngăn xếp lên 2 đơn vị. Tiếp theo nó sẽ chuyển byte thứ 2 và byte thứ 3 trong câu lệnh vào byte cao và byte thấp của <PC>. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

+ LCALL addr16 ; (PC) ← (PC)+3
(SP) ← (SP)+1
(SP) ← (PC7-0)
(SP) ← (SP)+1
(SP) ← (PC15-8)
(PC) ← (addr15-0)

Kích thước lệnh: Dài 3 byte.

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh: 0 0 0 1 0 0 1 0 addr15- addr8 addr7-addr0

Lệnh quay trở lại từ chương trình con

Cú pháp câu lệnh: RET

Chức năng: Lệnh RET được thực hiên ngay sau khi thực hiện xong lệnh ACALL và LCALL. Lệnh này chuyển byte cao và byte thấp từ ngăn xếp vào <PC> và giảm <SP> đi 2 đơn vị. Chương trình tiếp tục được thực hiện với lệnh có địa chỉ ở <PC>. Lệnh này không ảnh hưởng tới trạng thái của các cờ .

+ RET ; (PC15-8) ← ((SP))
(SP) ← (SP) -1
(PC7-0) ← ((SP))
(SP) ← (SP)-1

Kích thước lệnh: Dài 1 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 0 1 0 0 0 1 0

Lệnh quay trở lại từ ngắt

Cú pháp câu lệnh: RETI

Chức năng: Lệnh RETI chuyển byte cao và byte thấp từ ngăn xếp vào <PC>và để tiếp nhận các ngắt khác có cùng mức ưu tiên, sau đó giảm <SP> đi 2 đơn vị. Chương trình tiếp tục được thực hiện với lệnh trước khi xử lý ngắt. Lệnh này không ảnh hưởng tới trạng thái của các cờ.

+ RETI ; (PC15-8) ← ((SP))
(SP) ← (SP) -1
(PC7-0) ← ((SP))
(SP) ← (SP)-1

Kích thước lệnh: Dài 1 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 0 1 1 0 0 1 0

Lệnh nhảy gián tiếp

Cú pháp câu lệnh: JMP @ A+ DPTR

Chức năng: Lệnh JMP cộng giá trị không dấu 8 bit của thanh ghi tích luỹ với con trỏ dữ liệu 16 bit và nạp kết quả vào bộ đếm chương trình, kết quả này

chính là địa chỉ để nạp lệnh kế tiếp. Lệnh này không ảnh hưởng tới trạng thái các cờ.

+ JMP @A+DPTR ; (PC) ← ((A))+(DPTR)

Kích thước lệnh : Dài 1 byte.

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh : 0 1 1 1 0 0 1 1

Lệnh nhảy nếu một bit được thiết lập

Cú pháp câu lệnh: JB bit ,rel

Chức năng: Nếu bít đã cho có giá trị bằng 1 thì nó nhảy tới địa chỉ đã xác định, nếu không thì nó tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte thứ 3 của lệnh với nội dung trong <PC>. Lệnh này không ảnh hưởng tới trạng thái của các cờ .

+ JB bit, rel ; (PC) ← (PC) +3

IF(bit)=1

THEN

(PC) ← (PC) +rel

Kích thước lệnh: Dài 3 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 0 1 0 0 0 0 0 bit address reladdress

Lệnh nhảy nếu một bit không được thiết lập

Cú pháp câu lệnh: JNB bit ,rel

Chức năng: Nếu bit đã cho có giá trị bằng 0 thì nó nhảy tới địa chỉ đã xác định, nếu không thì nó tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte thứ 3 của lệnh với nội dung trong <PC>. Lệnh này không ảnh hưởng tới trạng thái của các cờ.

+ JNB bit,rel ; (PC) ← (PC) +3

IF(bit)=1

THEN

(PC) ← (PC)+rel

Kích thước lệnh: Dài 3 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 0 1 1 0 0 0 0 bit address reladdress

Lệnh nhảy nếu một bit được thiết lập và xoá bít đó

Cú pháp câu lệnh: JBC bit ,rel

Chức năng: Nếu bit đã cho có giá trị bằng 1 thì nó nhảy tới địa chỉ đã xác định và xoá bit này, nếu không thì nó tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte thứ 3 của lệnh với nội dung trong <PC>. Lệnh này không ảnh hưởng tới trạng thái của các cờ .

+ JBC bit,rel ; (PC) ← (PC) +3

IF(bit)=1

THEN

(bit) ← 0

(PC) ← (PC)+rel

Kích thước lệnh: Dài 3 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 0 0 1 0 0 0 0 bit address reladdress

Lệnh nhảy nếu cờ CF được thiết lập

Cú pháp câu lệnh: JC rel

Chức năng: Nếu cờ CF được thiết lập ở mức 1 thì nó nhảy tới địa chỉ đã xác định, nếu không thì nó tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte thứ 2 của lệnh với nội dung trong <PC>. Lệnh này không ảnh hưởng tới trạng thái của các cờ.

+ JNC rel ; (PC) ← (PC) +2

IF(C)=1

THEN

(PC) ← (PC)+rel

Kích thước lệnh: Dài 2 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 1 0 0 0 0 0 0 reladdress

Lệnh nhảy nếu cờ CF không được thiết lập

Cú pháp câu lệnh: JNC rel

Chức năng: Nếu cờ CF được thiết lập ở mức 0 thì nó nhảy tới địa chỉ đã xác định, nếu không thì nó tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte thứ 2 của lệnh với nội dung trong <PC. Lệnh này không ảnh hưởng tới trạng thái của các cờ.

+ JC rel; (PC) ← (PC) +2
IF(C)=0
THEN
(PC) ← (PC)+rel

Kích thước lệnh: Dài 2 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 1 0 1 0 0 0 0 rel address

Lệnh nhảy nếu thanh ghi tích luỹ bằng 0

Cú pháp câu lệnh : JZ rel

Chức năng: Nếu tất cả các bit của thanh ghi tích luỹ = 0 thì nó nhảy tới địa chỉ đã xác định, nếu không thì nó tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte thứ 2 của lệnh với nội dung trong <PC>. Lệnh này không ảnh hưởng tới trạng thái các cờ.

+ JZ rel ; (PC) ← (PC) +2
IF(A)=0
THEN
(PC) ← (PC)+rel

Kích thước lệnh: Dài 2 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 1 1 0 0 0 0 0 rel address

Lệnh nhảy nếu thanh ghi tích luỹ khác 0

Cú pháp câu lệnh: JNZ rel

Chức năng: Nếu có 1bit của thanh ghi tích lũy =1 thì nó nhảy tới địa chỉ đã xác định, nếu không thì nó tiếp tục thực hiện lệnh tiếp theo. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte thứ 2 của lệnh với nội dung trong <PC>. Lệnh này không ảnh hưởng tới trạng thái của các cờ.

+ JNZ rel ; (PC) ← (PC) +2
IF(A)=0
THEN
(PC) ← (PC)+rel

Kích thước lệnh: Dài 2 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 1 1 1 0 0 0 0 rel address

Lệnh nhảy khi so sánh 2 toán hạng

Cú pháp câu lệnh: CJNE <dest-byte>,<src-byte>, rel

Chức năng: Lệnh CJNE so sánh giá trị của 2 toán hạng đầu tiên, nếu 2 toán hạng không bằng nhau thì chương trình được rẽ nhánh. Địa chỉ rẽ nhánh được tính bằng các thêm rel với <PC>, sau khi đã tăng <PC> tới đầu của lệnh tiếp theo. Cờ <CF> sẽ được thiết lập nếu như giá trị nguyên không dấu của toán hạng đích nhỏ hơn giá trị nguyên không dấu của toán hạng nguồn, ngược lại thì cờ này bị xoá. Lệnh này không làm thay đổi giá trị các toán hạng.

+ CJNE A,direct,rel ; (PC) ← (PC) +3
IF (A)< >(direct)
THEN
(PC) ← (PC)+ liên quan ofset
IF(A) <(direcl)
THEN
(C) ← 1
ELSE
(C) ← 0

Kích thước lệnh: Dài 3 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 0 1 1 0 1 0 1 direct address rel address

+ CJNE A,#data , rel ; (PC) ← (PC) +3
IF (A)< >(data)
THEN
(PC) ← (PC)+rel
IF(A) <(data)
THEN
(C) ← 1
ELSE
(C) ← 0

Kích thước lệnh: Dài 3 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 0 1 1 0 1 0 0 immediate data rel address

+ CJNE Rn,#data , rel ; (PC) ← (PC) +3

IF(Rn) < >(data)

THEN

(PC) ← (PC)+liên quan ofset

IF(Rn) <(data)

THEN

(C) ← 1

ELSE

(C) ← 0

Kích thước lệnh: Dài 3 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh : 1 0 1 1 1 r r r immediate data rel address

+ CJNE @ Ri,#data , rel ; (PC) ← (PC) +3

IF(Ri) < >(data)

THEN

(PC) ← (PC)+liên quan ofset

IF(Rn) <(data)

THEN

(C) ← 1

ELSE

(C) ← 0

Kích thước lệnh: Dài 3 byte.

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 0 1 1 0 1 1 1 immediate data rel address

Lệnh giảm và nhảy

Cú pháp của câu lệnh : DJNZ byte , <rel-address>

Chức năng: Lệnh giảm ô nhớ đi 1 và nhảy tới địa chỉ cho bởi toán hạng thứ 2 nếu như kết quả khác 0, nếu như giá trị ban đầu là 00h thì nó chuyển qua 0FFh. Địa chỉ đích được tính bằng cách cộng thêm độ lệch có dấu trong byte lệnh cuối cùng với nội dung của <PC>, sau khi tăng <PC> tới byte dầu tiên của lệnh tiếp theo. Ngăn nhớ được giảm giá trị có thể là 1 thanh ghi hoặc 1 byte địa chỉ trực tiếp. Lệnh này không ảnh hưởng tới trạng thái của các cờ.

+ DJNZ Rn,rel ; (PC) ← (PC) +2
(Rn)) ← (Rn) –1
IF (Rn)>0 or (Rn)<0
THEN
(PC) ← (PC)+rel

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 1 0 1 1 r r r rel address

+ DJNZ direct,rel (PC) ← (PC) +2
(direct) ← (direct)-1
IF(direct)>0 or(direct)<0
THEN
(PC) ← (PC)+rel

Kích thước lệnh: Dài 3 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 1 0 1 0 1 0 1 direct data rel address

Lệnh tạm dừng hoạt động

Cú pháp của câu lệnh : NOP

Chức năng: Tạm dừng hoạt động chương trình khi có lệnh này và chương trình sẽ tiếp tục được thực hiện ở lệnh tiếp theo. Lệnh NOP không ảnh hưởng tới trạng thái các thanh ghi và các cờ hiệu.

+ NOP ; (PC) ← (PC)+1

Kích thước lệnh : Dài 1 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 0 0 0 0 0 0 0 0

10.3.4. Nhóm lệnh tính toán số học

Lệnh thực hiện phép cộng

Cú pháp của câu lệnh : ADD A, <src-byte>

Chức năng: Cộng giá trị biến với nội dung trong thanh ghi tích luỹ, kết quả lưu vào thanh ghi tích luỹ. Nếu có nhớ từ bit số 7 hoặc số 3 thì cờ nhớ <CF> hoặc cờ nhớ phụ <AF> được thiết lập và khi cộng các số nguyên không dấu mà bị tràn thì cờ nhớ <CF> cũng được thiết lập. Trường hợp thực hiện lệnh ADD mà có nhớ từ bit số 6 nhưng không nhớ từ bit số 7, hoặc có nhớ từ bit số 7 nhưng không nhớ

từ bit số 6 thì cờ <OV> sẽ được thiết lập, ngược lại thì cờ này bị xoá. Khi cộng các số nguyên có dấu mà tổng là một số âm thì cờ <OV>được thiết lập.

+ ADD A, Rn ; (A) ← (A) + (Rn)

Kích thước lệnh: dài 1byte

Thời gian thực hiện: 1 chu kỳ máy Mã lệnh: 0 0 1 0 1 r r r

+ ADD A,direct ; (A) ← (A) +(direct)

Kích thước lệnh : dài 2 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 0 1 0 0 1 0 1 direct address

+ ADD A , @(Ri) ; (A) ← (A) + ((Ri))

Kích thước lệnh: dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 0 1 0 0 1 1 1

+ ADD A, #data ; (A) ← (A) +#data

Kích thước lệnh: dài 2 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 0 1 0 0 1 0 0 immediate data

Lệnh cộng có nhớ

Cú pháp câu lệnh : ADDC A ,< src-byte>

Chức năng: Cộng giá trị biến với nội dung trong thanh ghi tích luỹ và cộng với cờ nhớ, kết quả lưu vào thanh ghi tích luỹ. Nếu có nhớ từ bit số 7 hoặc số 3 thì cờ nhớ <CF> hoặc cờ nhớ phụ <AF> được thiết lập và khi cộng các số nguyên không dấu mà bị tràn thì cờ nhớ <CF> cũng được thiết lập. Trường hợp thực hiện lệnh ADD mà có nhớ từ bit số 6 nhưng không nhớ từ bit số 7, hoặc có nhớ từ bit số 7 nhưng không nhớ từ bit số 6 thì cờ <OV> sẽ được thiết lập, ngược lại thì cờ này bị xoá. Khi cộng các số nguyên có dấu, mà tổng là một số âm thì cờ <OV>được thiết lập.

+ ADDC A, Rn ; (A) ← (A) + (Rn) +(C)

Kích thước lệnh: dài 1byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 0 1 1 1 r r r

+ ADDC A,direct ; (A) ← (A) +(direct)

Kích thước lệnh: dài 2 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 0 0 1 0 0 1 0 1 direct address

+ ADDC A , @Ri (A) ← (A) + ((Ri)) +(C)

Kích thước lệnh : dài 1byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 0 0 1 1 0 1 1 1

+ ADDC A , #data ; (A) ← (A) +(C) + #data

– Kích thước lệnh: dài 2 byte

– Thời gian thực hiện: 1 chu kỳ máy

– Mã lệnh: 0 0 1 1 0 1 0 0 immediate data

Lệnh trừ có nhớ

Cú pháp câu lệnh: SUBB A .< src-byte>

Chức năng: Lệnh SUBB thực hiện phép trừ thanh ghi tích lũy Acc cho toán hạng thứ 2 và cờ nhớ <CF>, kết quả lưu vào thanh ghi tích luỹ nếu kết quả phép trừ có nhớ thì cờ <CF> được thiết lập. Cờ nhớ phụ sẽ đươc thiết lập nếu có nhớ cho bit 3. Trường hợp thực hiện lệnh SUBB mà có nhớ từ bit số 6 nhưng không nhớ từ bit số 7, hoặc có nhớ từ bit số 7, nhưng không nhớ từ bit số 6 thì cờ <OV> sẽ được thiết lập, ngược lại thì cờ này bị xoá. Khi trừ các số nguyên có dấu, mà kết quả là một số âm thì cờ <OV> được thiết lập.

+ SUBB A , Rn ; (A) ← (A) – (C) –(Rn)

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 0 0 1 1 r r r

+ SUBB A,direct; (A) ← (A) – (C)-(direct)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 0 0 1 0 1 0 1 direct address

+ SUBB A, @ Ri (A) ← (A) – (C)-((Ri))

Kích thước lệnh : Dài 1 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 1 0 0 1 0 1 1 1

+ SUBB A;#data (A) ← (A) – (C)-(direct)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 1 0 0 1 0 1 0 0 immediate data

Lệnh tăng lên một đơn vị

Cú pháp của câu lệnh: INC <byte>

Chức năng : Lệnh INC tăng giá trị của biến đã cho lên 1 đơn vị. Biến có độ dài 1 byte do đó nếu giá trị ban đầu là 0FFh khi tăng lên 1 thì kết quả là 00h. Lệnh này không ảnh hưởng tới trạng thái các cờ.

+ INC A ; (A) ← (A) +1

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 0 0 0 0 0 1 0 0

+ INC dirrect , dirrect ← (direct)+1

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 0 0 0 0 0 1 0 1 dirrect address

+ INC @Ri ; (Ri) ← (Ri)+1

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 0 0 0 0 1 1 1

+ INC Rn ; (Rn) ← (Rn)+1

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 0 0 0 1 r r r

Lệnh giảm 1 đơn vị

Cú pháp của câu lệnh: DEC <byte>

Chức năng: Lệnh DEC giảm giá trị của biến đã cho xuống 1 đơn vị. Biến có độ dài 1 byte do đó nếu giá trị ban đầu là 00h khi giảm 1 thì kết quả là FFh. Lệnh này không ảnh hưởng tới trạng thái các cờ.

+ DEC A ; (A) ← (A)-1

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 0 0 0 1 0 1 0 0

+ DEC direct; Dirrect ← (Direct)-1

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh : 0 0 0 1 0 1 0 1 Dirrect address

+ DEC @Ri ; (Ri) ← (Ri)-1

  Kích thước lệnh: Dài 1 byte

  Thời gian thực hiện: 1 chu kỳ máy

  Mã lệnh: 0 0 0 0 0 1 1 1

+ DEC Rn ; (Rn) ← (Rn)-1

  Kích thước lệnh: Dài 1 byte

  Thời gian thực hiện: 1 chu kỳ máy

  Mã lệnh: 0 0 0 1 1 r r r

Lệnh tăng con trỏ dữ liệu

Cú pháp của câu lệnh: INC DPTR

Chức năng: Tăng con trỏ dữ liệu 16 bit lên 1 khi byte thấp của con trỏ dữ liệu <DPL>bị tràn, làm giá trị của nó từ 0FFh chuyển sang 00h thì byte cao của con trỏ dữ liệu <DPH>tăng thêm 1 đơn vị. Lệnh này không ảnh hưởng tới trạng thái các cờ.

+ INC DPTR ; (DPTR) ← (DPTR)+1

  Kích thước lệnh: Dài 1 byte

  Thời gian thực hiện: 2 chu kỳ máy

  Mã lệnh: 1 0 1 0 0 0 1 1

Lệnh thực hiện phép nhân

Cú pháp của câu lệnh: MUL AB

Chức năng: Thực hiện phép nhân các số nguyên 8 bit không dấu trong thanh ghi tích luỹ với thanh ghi B, byte thấp của kết quả 16 bit được lưu trong thanh ghi tích luỹ, còn byte cao thì lưu trong thanh ghi B. Nếu kết quả lớn hơn 0FFh thì cờ tràn <OF> sẽ được thiết lập, cờ <CF> luôn bị xoá.

+ MUL AB ; (A7-0) ,(B 15-8) ← (A)*(B)

  Kích thước lệnh: Dài 1 byte

  Thời gian thực hiện: 4 chu kỳ máy

  Mã lệnh: 1 0 1 0 0 1 0 0

Lệnh thực hiện phép chia

Cú pháp của câu lệnh: DIV AB

Chức năng: Thực hiện phép chia các số nguyên 8bit không dấu trong thanh ghi tích luỹ cho thanh ghi B, phần nguyên của thương số được lưu trong thanh ghi tích luỹ, còn phần dư thì lưu trong thanh ghi B. Cờ <OF>và cờ <CF> sẽ bị xoá .

+ DIV AB ; (A7-0) ,(B 15-8) ← (A)/(B)

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 4 chu kỳ máy

Mã lệnh: 1 0 0 0 0 1 0 0

Lệnh hiệu chỉnh số thập phân

Cú pháp của câu lệnh : DA A

Chức năng: Hiệu chỉnh 8 bit kết quả sau khi thực hiện của phép cộng lưu trong thanh ghi tích luỹ Acc. Nếu 4 bit thấp trong <Acc> lớn hơn 9 hoặc <CF=1> thì phải cộng thêm 6 vào <Acc> để cho chữ số dạng thập phân được chính xác trong nửa byte thấp. Nếu cờ CF đã được thiết lập, hoặc 4 bit cao có giá trị vượt quá 9 thì cũng phải cộng thêm 6 vào 4 bit cao.

+ DA ; If {[(A3-0) >9] v[(CF)=1]}

THEN (A3-0) ← (A3-0)+6

AND

If {[(A7-4) >9] v[(CF)=1]}

THEN (A7-4) ← (A7-4)+6

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 1 1 0 1 0 1 0 0

10.3.5. Nhóm lệnh tính toán logic

Lệnh AND cho các biến 1 byte

Cú pháp của câu lệnh: ANL ,<dest-byte>,<src-byte>

Chức năng: Lệnh ANL thực hiện phép tính logic AND theo mức bit giữa các biến dài 1 byte đã cho. Kết quả lưu vào toán hạng đích (toán hạng thứ nhất). Toán hạng nguồn cho phép 6 chế độ địa chỉ hoá, khi toán hạng đích là thanh ghi tích luỹ thì toán hạng nguồn có thể là 1 thanh ghi, 1 ô nhớ trực tiếp hay gián tiếp hoặc là địa chỉ trực tiếp. Khi toán hạng đích là địa chỉ trực tiếp thì toán hạng nguồn có thể là thanh ghi tích lũy hoặc số liệu trực tiếp. Lệnh này không ảnh hưởng tới trạng thái các cờ.

+ ANL A, Rn ; (A) ← (A) ^ (Rn)

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 1 0 1 1 r r r

+ ANL A, (A) ← (A) ^ (Direct)

Kích thước lệnh : Dài 2 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 0 1 0 1 0 1 0 1 Dirrect address

+ ANL A @ Ri ; (A) ← (A) ^ (Ri)

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 1 0 1 0 1 1 1

+ ANL A, #data ; (A) ← (A) ^ #data

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0 1 0 1 0 1 0 0 immediate data

+ ANL direct, A ; (A) ← (direct) ^ (A)

Kích thước lệnh : Dài 2 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh : 0 1 0 1 0 0 1 0 direct address

+ ANL direct, #data ; (direct) ← (direct)^ #data

Kích thước lệnh: Dài 3 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0 1 0 1 0 0 1 1 direct address immediate data

Lệnh AND cho các biến 1 bit

Cú pháp của câu lệnh: ANL C , <src-bit>

Chức năng: Thực hiện phép tính logic AND cho các biến mức bit. Nếu giá trị logic của toán hạng nguồn bằng 0 thì cờ <CF> bị xoá. Dấu "/" đứng trước một toán hạng cho biết phần bù logic của bít đã được định địa chỉ và được sử dụng như toán hạng nguồn. Chỉ có địa chỉ trực tiếp được dùng làm toán hạng nguồn. Lệnh này không làm ảnh hưởng tới trạng thái các cờ khác.

+ ANL C, bit ; (C) ← (C)^(bit)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 0 0 0 0 0 1 0 bit địa chỉ

+ ANL C, /bit ; (C) ← (C) ^ (bit)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 0 1 1 0 0 0 0 Bit địa chỉ

Lệnh OR cho các biến 1 byte

Cú pháp của câu lệnh: ORL <dest-byte>, <src-byte>

Chức năng: Lệnh ORL thực hiện phép tính logic OR mức byte giữa các biến đã cho, kết quả được lưu trong toán hạng đích. Trong lệnh này hai toán hạng cho phép 6 tổ hợp chế độ định địa chỉ. Khi toán hạng đích là thanh ghi tích luỹ thì toán hạng nguồn có thể là thanh ghi, địa chỉ trực tiếp, gián tiếp hoặc giá trị trực hằng. Khi toán hạng nguồn là địa chỉ trực tiếp thì nó có thể là thanh ghi tích luỹ hay giá trị trực hằng. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

+ ORL A, Rn ; (A) ← (A)v (Rn)

Kích thước lệnh : Dài 1 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 0100 1rrr

+ ORL A, drect ; (A) ← (A)v(direct)

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0101 0101 direct address

+ ORL A, @Ri ; (A) ← (A)v ((Ri))

Kích thước lệnh :Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0100 0111

+ ORL A, #data ; (A) ← (A)v#data

Kích thước lệnh: dài 2byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 0100 0100 immediate data

+ ORLdrect, A ; (direct) ← (direct) v (A)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 0100 0100 direct address

+ ORLdirect, data ; (direct) ← (direct) v #data

Kích thước lệnh: Dài 3 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0100 0011 direct address immediate data

Lệnh ORL cho các biến một bit

Cú pháp của câu lệnh : ORL C, <src-bit >

Chức năng: Thực hiện phép tính logic OR cho các biến mức bit. Nếu giá trị lo gic của toán hạng nguồn bằng 1 thì cờ <CF>được thiết lập. Dấu "/" đứng trước một toán hạng cho biết phần bù logic của bit đã được định địa chỉ và được sử dụng như toán hạng nguồn, chỉ có địa chỉ trực tiếp được dùng làm toán hạng nguồn. Lệnh này không làm ảnh hưởng tới trạng thái các cờ khác.

+ ORL C, bit ; (C) ← (C) v (bit)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện : 2 chu kỳ máy

Mã lệnh: 0111 0 0 10 bit địa chỉ

+ ORL C, /bit ; (C) ← (C)v(bit)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 1 0 10 0 0 0 0 Bit địa chỉ

Lệnh XOR cho các biến một byte

Cú pháp của câu lệnh : XRL <dest-byte>, <src-byte >

Chức năng: Lệnh XRLthực hiện phép tính logic XOR mức byte giữa các biến đã cho, kết quả được lưu trong toán hạng đích. Trong lệnh này hai toán hạng cho phép 6 tổ hợp chế độ định địa chỉ, khi toán hạng đích là thanh ghi tích luỹ thì toán hạng nguồn có thể là thanh ghi, địa chỉ trực tiếp, gián tiếp hoặc giá trị trực hằng, khi toán hạng nguồn là địa chỉ trực tiếp thì toán hạng nguồn có thể là thanh ghi tích lũy hay giá trị trực hằng. Lệnh này không làm ảnh hưởng tới trạng thái các cờ.

+ XRL A, Rn ; (A) ← (A) ⊕ (Rn)

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0110 1rrr

+ XRL A, direct ; (A) ← (A) ⊕ (direct)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0110 0101 direct address

+ XRL A, @Ri (A) ← (A) ⊕ ((Ri))

Kích thước lệnh: Dài 1 byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh : 0110 0111

+ XRL A, #data ; (A) ← (A) ⊕ #data

Kích thước lệnh: dài 2byte

Thời gian thực hiện: 1 chu kỳ máy

Mã lệnh: 0110 0100 immediate data

+ XRL drect , A ; (direct) ← direct) ⊕ (A)

Kích thước lệnh: Dài 2 byte

Thời gian thực hiện : 1 chu kỳ máy

Mã lệnh: 0110 0100 direct address

+ XRLdirect , #data ; (direct) ← (direct) ⊕ #data

Kích thước lệnh: Dài 3 byte

Thời gian thực hiện: 2 chu kỳ máy

Mã lệnh: 0110 0011 direct address immediate data

10.4. TÓM TẮT TẬP LỆNH CỦA HỆ VI XỬ LÝ ON-CHIP 80C51

Các lệnh số học (Arithmetic Instrustion):

ADD A, <src, byte>

ADD A, Rn : (A) ← (A) + (Rn)

ADD A, direct : (A) ← (A) + (direct)

ADD A, @ Ri : (A) ← (A) + ((Ri))

ADD A, # data : (A) ← (A) + # data

ADDC A, Rn : (A) ← (A) + (C) + (Rn)

ADDC A, direct : (A) ← (A) + (C) + (direct)

ADDC A, @ Ri : (A) ← (A) + (C) + ((Ri))

ADDC A, # data : (A) ← (A) + (C) + # data

SUBB A, <src, byte>

SUBBA, Rn : (A) ← (A) - (C) - (Rn)

SUBBA, direct : (A) ← (A) - (C) - (direct)

SUBBA, @ Ri : (A) ← (A) - (C) - ((Ri))

SUBBA, # data : (A) ← (A) - (C) - # data

INC <byte>

INC A : (A) ← (A) + 1

INC direct: (direct) ← (direct) + 1

INC Ri : ((Ri)) ← ((Ri)) + 1

INC Rn : (Rn) ← (Rn) + 1

INC DPTR: (DPTR) ← (DPTR) + 1

DEC <byte>

DEC A : (A) ← (A) - 1

DEC direct : (direct) ← (direct) - 1

DEC @Ri : ((Ri)) ← ((Ri)) - 1

DEC Rn : (Rn) ← (Rn) - 1

MULL AB : (A) ← LOW [(A) x (B)]

: (B) ← HIGH [(A) x (B)]

DIV AB : (A) ← Integer Result of [(A)/(B)]; OV

: (B) ← Remainder of [(A)/(B)];

DA A :[(A3-A0)>9] [(AC)=1] ⇐ (A3÷A0) ← (A3÷A0) + 6.

[(A7-A4)>9] [(C)=1] ⇐ (A7÷A4) ← (A7÷A4) + 6.

ANL <dest - byte> <src - byte>

ANL A, Rn : (A) ← (A) AND (Rn).

ANL A, direct : (A) ← (A) AND (direct).

ANL A,@ Ri : (A) ← (A) AND ((Ri)).

ANL A, # data : (A) ← (A) AND (# data).

ANL direct, A : (direct) ← (direct) AND (A).

ANL direct, # data : (direct) ← (direct) AND # data.

ORL <dest - byte> <src - byte>

ORL A, Rn : (A) ← (A) OR (Rn).

ORL A, direct : (A) ← (A) OR (direct).

ORL A,@ Ri : (A) ← (A) OR ((Ri)).

ORL A, # data : (A) ← (A) OR # data.

ORL direct, A : (direct) ← (direct) OR (A).

ORL direct, # data : (direct) ← (direct) OR # data.

XRL <dest - byte> <src - byte>

XRL A, Rn : (A) ← (A)) (Rn).

XRL A, direct : (A) ← (A)) (direct).

XRL A,@ Ri : (A) ← (A)) ((Ri)).

XRL A, # data : (A) ← (A)) # data.

XRL direct, A : (direct) ← (direct)) (A).

XRL direct, # data : (direct) ← (direct)) # data.

CLR A : (A) ← 0

CLR C : (C) ← 0

CLR Bit : (Bit) ← 0

RL A : (An + 1) ← (An); n = 0÷6;(A_0) ← (A_7)

RLC A : (An + 1) ← (An); n = 0÷6;(C) ← (A_7)
(A_0) ← (C)

RR A : (An + 1) → (An); n = 0÷6; A_0) → (A_7)

RRC A : (An + 1) → (An); n = 0÷6; (C) → (A_7)
(A_0) → (C)

SWAP A : (A3÷A0)√(A7÷A4).

CÁC LỆNH NHẢY:

JC rel :

JNC rel :

JB bit, rel :

JNB bit, rel :

JBC bit, rel :

ACALL addr11 : (PC) ← (PC) + 2;(SP) ← (SP) + 1
((SP)) ← (PC7÷PC0);(SP) ← (SP) + 1
((SP))←(PC15÷PC8);(PC10÷PC0)←page Address.

LCALL addr16 : (PC) ← (PC) + 3;(SP) ← (SP) + 1
((SP)) ← (PC7÷PC0);(SP) ← (SP) + 1
((SP)) ← (PC15÷PC8);(PC) ← Addr15÷Addr0.

RET : (PC15÷PC8) ← (SP);(SP) ← (SP) - 1
(PC7÷PC0) ← ((SP));(SP) ← (SP) -1.

RETI

AJMP Addr11 : (PC) ← (PC) + 2;(PC10÷PC0)←page Address.

LJMP Addr16 :

SJMP rel : (PC) ← (PC) + 2; (PC) ← (PC) + byte 2

JMP @ A + DPTR : (PC) ← (A) + (DPTR)

JZ rel : (PC) ← (PC) + 2;(A) = 0 ⇐ (PC) ← (PC) + byte 2

JNZ rel : (PC) ← (PC) + 2;(A) < > 0 ⇐ (PC) ← (PC) + byte 2

CJNE A, direct, rel : (PC) ← (PC) + 3

(A) < > (direct) ⇐ (PC) ← (PC) + Relative Address.

(A) < (direct) ⇐ C = 1;(A) > (direct) ⇐ C = 0

(A) = (direct).

CJNE A, # data, rel :

CJNE Rn, # data, rel :

CJNE @ Ri, # data, rel :

DJNE Rn, rel : (PC) ← (PC) + 2;(Rn) ← (Rn) -1

(Rn) < > 0 ⇐ (PC) ← (PC) + byte 2.

DJNZ direct, rel :

CÁC LỆNH CHUYỂN DỮ LIỆU:

MOV A,Rn : (A) ← (Rn)

MOV A, direct : (A) ← (direct)

MOV A, @ Ri : (A) ← ((Ri))

MOV A, # data : (A) ← # data

MOV Rn, A : (Rn) ← (A)

MOV Rn, direct : (Rn) ← (direct)

MOV Rn, # data : (Rn) ← # data

MOV direct, A : (direct) ← (A)

MOV direct, Rn : (direct) ← (Rn)

MOV direct, direct: (direct) ← (direct)

MOV direct, @ Ri : (direct) ← ((Ri))

MOV direct, # data: (direct) ← data

MOV @ Ri, A : ((Ri)) ← (A)

MOV @ Ri, direct : ((Ri)) ← (direct)

MOV @ Ri, # data : ((Ri)) ← # data

MOV DPTR, # data16: (DPTR) ← # data16

MOV A, @ A + DPTR: (A) ← (A) + (DPTR)

MOV @ A + PC : (PC) ← (PC) + 1; (A) ← (A) + (PC)

MOVX A, @ Ri : (A) ← ((Ri))

MOVX A, @ DPTR: (A) ← ((DPTR))

MOVX @ Ri, A : ((Ri)) ← (A)

MOVX @ DPTR, A: ((DPTR)) ← (A)

PUSH direct : (SP) ← (SP) + 1
(SP) ← (Drirect)

POP direct : (direct) ← ((SP))
(SP) ← (SP) - 1

XCH A, Rn : (A) √ (Rn)

XCH A, direct : (A) √ (direct)

XCH A, @ Ri : (A) √ ((Ri))

XCHD A, @ Ri : (A3÷A0) √ ((Ri3÷Ri0))

<u>**CÁC LỆNH XỬ LÝ MỨC BIT:**</u>

CLR C :

CLR BIT :

SET C :

SET BIT :

CPL C :

CPL BIT :

ANL C, BIT: (C) ← (C) AND (BIT)

ANL C, /BIT: (C) ← (C) AND NOT (BIT

ORL C, BIT: (C) ← (C) OR (BIT) :

ORL C, /BIT: (C) ← (C) OR NOT (BIT) :

MOV C, BIT: (C) ← (BIT) :

MOV BIT, C: (BIT) ← (C) :

CHƯƠNG 11
THIẾT KẾ HỆ XỬ LÝ TRÊN HỆ VI XỬ LÝ ON-CHIP 80C51

Khi thực hiện vấn đề thiết kế hệ vi xử lý chuyên dụng trên bộ vi xử lý độc lập, ta đã nói tới tính ưu việt của hệ như có tính mềm dẻo cao trong thao tác, có tốc độ cao so với hệ đa dụng do chức năng được chuyên năng hoá cao, có độ tin cậy làm việc cao do các mạch vi điện tử IC sử dụng trong hệ là các IC có mức tổ hợp cao LSI hoặc cực cao VLSI và có thể dễ dàng thay đổi thay đổi thông số, trình tự vận hành kể cả thay đổi chức năng của hệ thống chỉ bằng cách thay đổi phần mềm cài đặt bên trong hệ thống mà không phải thay đổi phần cứng của hệ.

Khi thực hiện vấn đề thiết kế hệ vi xử lý chuyên dụng trên On-chip, các ưu điểm trên vẫn được giữ nguyên nhưng nó có tính vượt trội về mặt độ tin cậy và đặc biệt là sự thuận lợi trong tổ chức phần cứng. Nhờ có nhiều thành phần chức năng đã chứa sẵn trong On-chip như Rom, Ram, Timer, cổng vào/ra ... mà cấu trúc phần cứng của hệ vi xử lý cần thiết kế trở nên rất gọn và đơn giản. Ưu điểm này có lẽ là ưu điểm lớn nhất của hệ vi xử lý trên On-chip, vì vậy On-chip trở thành phổ cập và nó có mặt ở hầu hết trong các ứng dụng của kỹ thuật hiện đại.

11.1. TRÌNH TỰ THIẾT KẾ HỆ VI XỬ LÝ CHUYÊN DỤNG TRÊN ON-CHIP

Khi thiết kế hệ vi xử lý trên On-chip cần tuân thủ các bước theo trình tự sau:

Bước 1. Phân tích chức năng, nhiệm vụ hệ vi xử lý cần thiết kế.

Nhiệm vụ của hệ vi xử lý cần thiết kế bao giờ cũng được mô tả đầy đủ bằng các ngôn ngữ kỹ thuật thích hợp. Người thiết kế phải tiến hành nghiên cứu và phân tích một cách kỹ lưỡng nhiệm vụ và các chức năng chính của hệ vi xử lý. Tìm hiểu và nghiên cứu môi trường làm việc của hệ, đối tượng điều khiển của hệ, đặc trưng và tham số của nguồn thông tin mà hệ cần thu nhận và xử lý v.v... Trên cơ sở của bước phân tích phải phân chia một cách hợp lý chức năng nào thuộc phần cứng đảm nhiệm và những chức năng nào do phần mềm đảm nhiệm.

Phần cứng, trước hết là những thành phần không thể thay thế bằng phần mềm như các cấu trúc vật lý của các thiết bị ngoại vi cần được ghép với On-chip ... và một số chức năng của phần mềm mà muốn cải thiện tính năng nào đó như về tốc độ chẳng hạn khi phải thực hiện các phép tính với các hàm toán học phức tạp.

Phần mềm đảm nhiệm các chức năng còn lại của hệ thống để thực hiện các thao tác phức tạp đòi hỏi phải xử lý tình huống, ra quyết định trong các điều kiện và yếu tố tác động vào hệ là các điều kiện và yếu tố động (luôn biến đổi).

Bước 2. Tổ chức phần cứng cho hệ vi xử lý cần thiết kế.

Trên cơ sở của các phân tích của bước 1, ở bước 2 phải tổ chức được phần cứng của hệ thống. Các nội dung cần thực hiện ở bước này là:

Xây dựng sơ đồ khối của hệ vi xử lý cần thiết kế, trong đó mỗi thành phần được thể hiện bằng một hộp chức năng. Các thành phần này liên hệ với On-chip nhau thông qua kênh hệ thống của hệ, đó là kênh địa chỉ, kênh dữ liệu và kênh điều khiển.

Tiến hành lựa chọn các chip IC phù hợp với chức năng nhiệm vụ của từng thành phần. Quan trọng nhất là chọn chủng loại On-chip vì nó sẽ quyết định tới khả năng hoạt động chung của hệ như tốc độ, độ rộng kênh dữ tiệu, khả năng quản lý không gian nhớ và quản lý các thiết bị ngoại vi. Nếu hệ có số lượng ngoại vi lớn phải tổ chức các bộ đệm kênh để tránh ảnh hưởng tới tham số của kênh hệ thống như hiện tượng quá tải kênh. Các bộ đệm kênh hệ thống nên sử dụng dạng đệm 3 trạng thái vì nó cho phép bảo vệ kênh tốt nhất. Trong một số trường hợp cụ thể, khi số lượng ngoại vi ít, có thể nối thẳng không qua bộ đệm kênh nhằm tối giản cấu trúc phần cứng.

Xây dựng các sơ đồ nối ghép chi tiết cho từng thành phần có tính tới cơ chế điều khiển và cơ chế đồng bộ sao cho thật tối ưu nhằm tránh hiện tương xung đột trạng thái của hệ. Cụ thể:

Tổ chức bộ nhớ trung tâm có dung lượng và cơ chế xuất nhập thông tin đúng theo yêu cầu. Không gian nhớ mà các chip IC bộ nhớ có thể là không gian nhớ nội trú hoặc ngoại trú hoặc có thể kết hợp cả hai loại hình trên. Cần lưu ý khi tổ chức tín hiệu kích hoạt chip Chip Select và điều khiển hướng truyền thông tin sao cho phải đơn trị không trùng lặp.

Tổ chức các ngoại vi cần thiết theo đúng tính năng của từng ngoại vi. Khi thiết kế các ngoại vi này điều quan tâm hơn cả là khả năng và phương thức giao tiếp của chúng với hệ vi xử lý On-chip. Có các phương thức giao tiếp có thể sử dụng là: hỏi vòng, ngắt hoặc phương thức truy nhập trực tiếp. Mỗi phương thức có ưu nhược điểm riêng nên người thiết kế phải biết lựa chọn cho đúng với điều kiện và nhiệm vụ cụ thể của hệ mà sử dụng phương thức này hay phương thức khác.

Tổ chức bộ giải mã địa chỉ chọn chip, tức là gán cho mỗi ngoại vi một địa chỉ riêng biệt mà bộ vi xử lý sẽ sử dụng để kích hoạt các thành phần thuộc quyền quản lý của mình nhằm thực hiện các thao tác cần thiết.

Nhiệm vụ cuối cùng là tổ chức ghép nối tất cả các thành phần với On-chip thông qua kênh dữ liệu, địa chỉ và hệ thống để tạo thành hệ hoàn chỉnh, sẵn sàng đi vào hoạt động khi có chương trình MONITOR cài đặt bên trong hệ vi xử lý.

Bước 3. Xây dựng phần mềm cho hệ vi xử lý cần thiết kế.

Trên cơ sở của các phân tích của bước 1, bước 2, ở bước 3 phải xây dựng phần mềm của hệ thống. Các nội dung cần thực hiện ở bước này là:

Xây dựng thuật toán điều khiển bảo đảm quản lý và phối hợp các chức năng của hệ thật tối ưu như phân phối mức ưu tiên của các chức năng, chỉ rõ các giới hạn của từng chức năng nhằm làm cơ sở để tổ chức lưu đồ thuật toán ở bước kế tiếp.

Xây dựng lưu đồ thuật toán tổng quát cho hoạt động của hệ vi xử lý. Kế đến là các lưu đồ thuật toán cho các modul chức năng nhằm cụ thể hoá một cách chi tiết các trình tự thao tác của hệ thống.

Viết chương trình nguồn. Thường thì nên viết dưới dạng file dạng COM là dạng gói gọn trong mảng nhớ <=64 kb. Dạng file nguồn này cho phép chương trình MONITOR phản ứng nhanh với các tác động và yêu cầu.

Bước 4. Nạp chương trình cho hệ VXL On-chip cần thiết kế.

Trên cơ sở của các phân tích của bước 1, bước 2 và bước 3, ở bước 4 phải nạp chương trình cho hệ vi xử lý cần thiết kế theo các cách sau:

Nếu chỉ sử dụng ROM nội trú phải có thiết bị nạp chương trình MONITOR riêng. Nếu chỉ sử dụng ROM ngoại trú chỉ cần có thiết bị nạp phổ dụng. Quá trình chạy thử người thiết kế phải theo dõi để tiến hành điều chỉnh cho tới khi hệ hoạt động hoàn toàn thoả mãn các yêu cầu đặt ra. Quá trình nạp ROM có thể lặp lai nhiều lần vì mỗi lần sửa chương trình lại phải dịch và nạp lại.

11.2. THIẾT KẾ CÁC HỆ VI XỬ LÝ CHUYÊN DỤNG

Hệ chức năng 1. Thiết kế hệ đo độ rộng xung

Thiết kế hệ đo độ rộng xung trên hệ vi xử lý on-chip 80C51 với *focs*= 12 MHz, xung cần đo độ rộng được đưa vào chân tín hiệu INT0 (bit P3.2 – chân IC 12), kết quả đo phải chứa trong hai thanh ghi: R1 chứa byte cao của kết quả, R0 chứa byte thấp của kết quả.

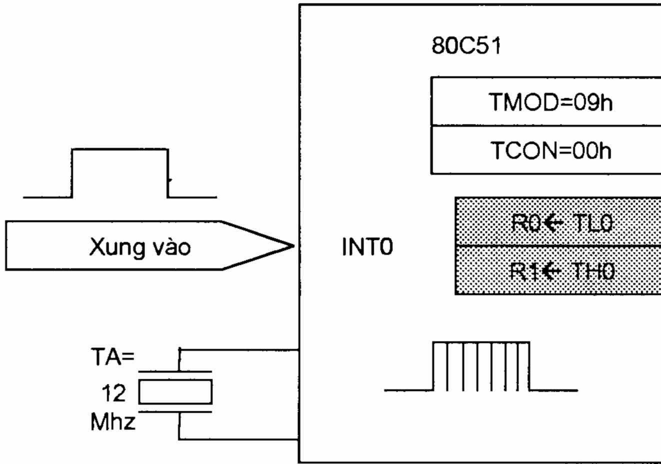
Phân tích: Căn cứ vào đầu bài thì hệ thống cần thiết kế có dạng như mô tả trên hình 11.1. Phương pháp đo là chèn một chuỗi xung (xung răng lược) có độ rộng xác định trước trong thời gian tồn tại của xung cần đo.

*Độ rộng xung cần đo = (chu kỳ xung răng lược) * (số lượng xung răng lược).*

Sai số của phép đo max = | 1/2 *chu kỳ xung răng lược* |.

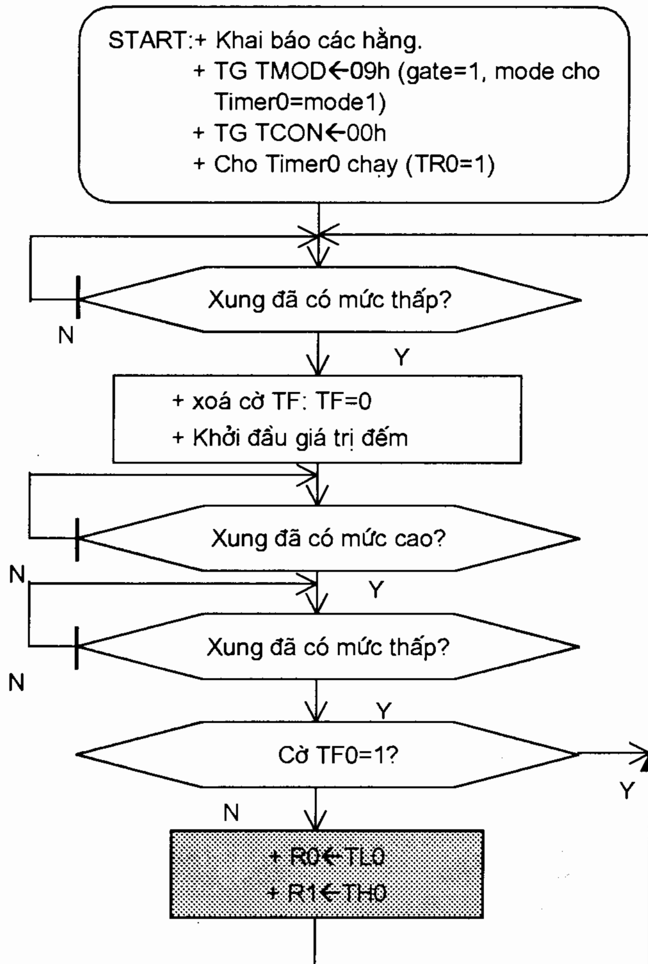
Vì tần số dao động nội = 12 MHz/12 = 1 MHz nên chu kỳ xung răng lược = 1 micro sec nên nội dung Timer0 chính là kết quả. Sử dụng Timer0 chạy ở mode 1 để tạo xung răng lược. Muốn vậy, thanh ghi TMOD = 09h = 0000 1001 (gate0 = 1 tạo điều kiện cho INT0 được kích hoạt Timer0, C/T0 = 0 làm việc với dao động nội, mode cho Timer0 = mode1).

Khởi đầu cần xoá cờ TF0 và cấm Timer0 chạy TR0 = 0. Do đó TCON = 00h = 0000 0000 (TF0 = TR0 = 0).



Hình 11.1. **Sơ đồ hệ đo độ rộng xung trên hệ VXL on chip**

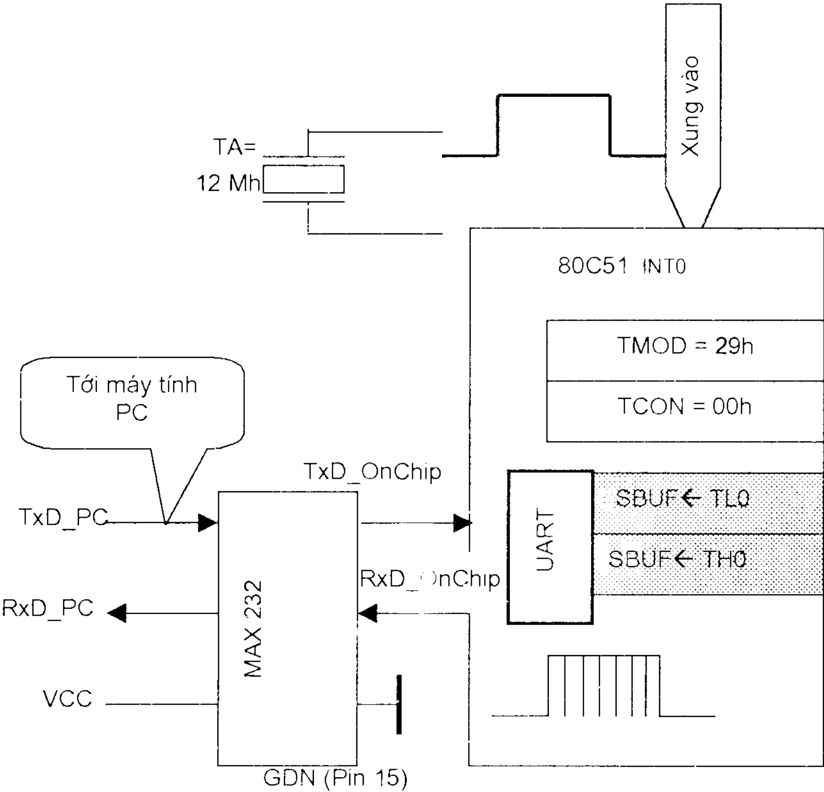
Lưu đồ thuật toán: Từ phân tích trên, lưu đồ thuật toán cho chương trình có dạng:



Vì tần số dao động nội= 12MHZ/12= 1 MHZ nên chu kỳ xung răng lược= 1 micro sec nên nội dung Timer0 chính là kết quả. Do vậy, sử dụng Timer0 chạy ở mode 1 để tạo xung răng lược, Timer1 chạy ở mode 2 để tạo tốc độ baud cho UART. Muốn vậy, thanh ghi TMOD = 29h = 0010 1001 (gate1 = 0, C/T1 = 0, mode cho Timer1= mode2 – tự nạp lại hệ số chia; gate0 = 1 tạo điều kiện cho INT0 được kích hoạt Timer0, C/T0 = 0 chạy ở chế độ đồng hồ, mode cho Timer0 = mode1).

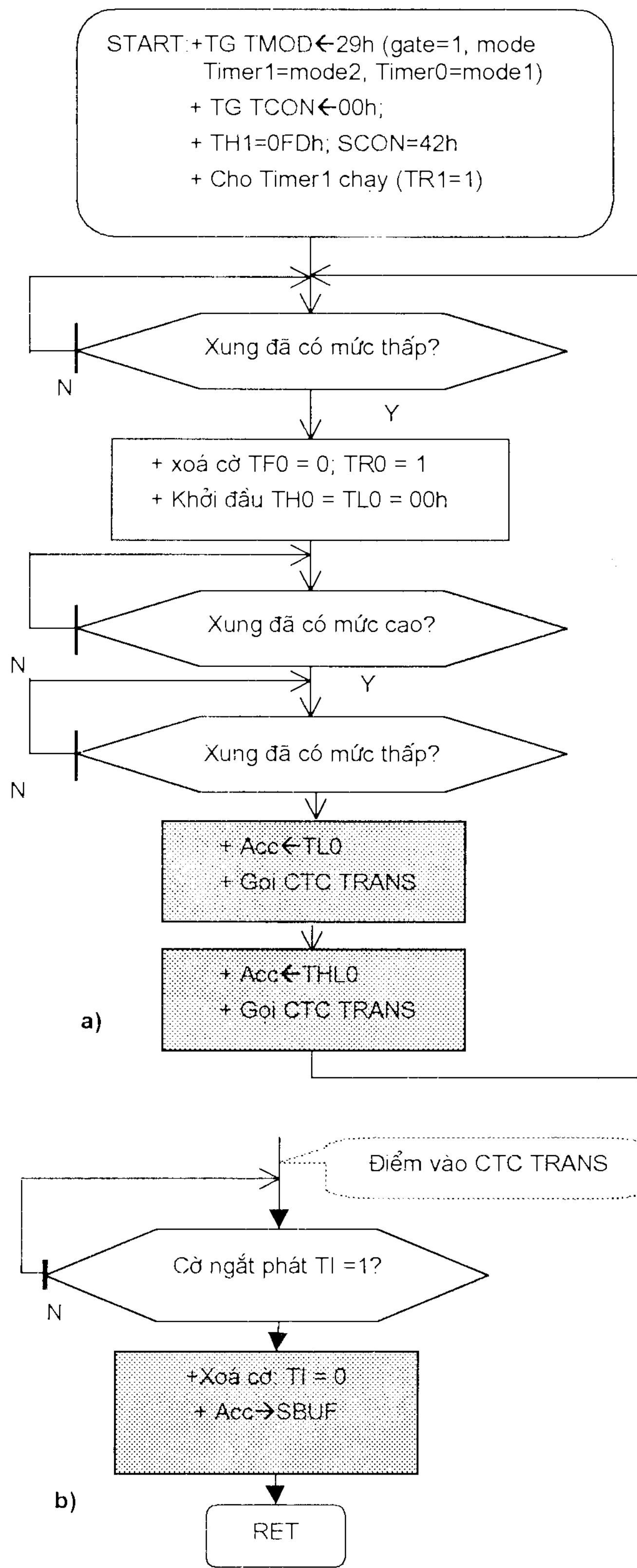
Khởi đầu cần xoá cờ TF0, TF1 và cấm Timer0, Timer1 chạy TR0 = TR1 = 0. Do đó TCON = 00h = 0000 0000 (TF1 = TR1 = TF0 = TR0 = 0).

SCON = 42h = 0100 0010 – Mode1 cho UART, máy phát (bit N1- cờ TI = 1) sẵn sàng, cấm thu (bit N4- cờ REN = 0).



Hình 11.2. Sơ đồ hệ đo độ rộng xung trên hệ VXL on chip ghép với PC.

Lưu đồ thuật toán: Từ phân tích trên, lưu đồ thuật toán cho chương trình có dạng:


START: +TG TMOD←29h (gate=1, mode Timer1=mode2, Timer0=mode1)
+ TG TCON←00h;
+ TH1=0FDh; SCON=42h
+ Cho Timer1 chạy (TR1=1)
Xung đã có mức thấp?
N
Y
+ xoá cờ TF0 = 0; TR0 = 1
+ Khởi đầu TH0 = TL0 = 00h
Xung đã có mức cao?
N
Y
Xung đã có mức thấp?
N
+ Acc←TL0
+ Gọi CTC TRANS
+ Acc←THL0
+ Gọi CTC TRANS
a)
Điểm vào CTC TRANS
Cờ ngắt phát TI =1?
N
+Xoá cờ TI = 0
+ Acc→SBUF
b)
RET

Viết chương trình: Dựa theo lưu đồ thuật toán, chương trình Assembly có dạng:

```
PUSLSE  EQU    INT0
    ORG    2000H
INIT:
MOV     TMOD,#29H ; gate1=0, C/T1=0, mode cho Timer1= mode2 –
                  ;tự nạp lại  hệ số chia; gate0=1 tạo điều kiện cho INT0
                  ;được kích hoạt Timer0, C/T0=0 chạy ở chế độ
                  ;đồng hồ, mode cho Timer0=  mode1
 MOV     TH1,#0FDH  ; tốc độ = 9600 baud
MOV     TCON,#00H
MOV     SCON,#42H;  Mode1 cho UART, máy phát (bit N1- cờ TI= 1)
                    ; sẵn sàng, cấm thu (bit N4- cờ REN= 0).
    SETB     TR1  ; cho Timer1 chạy ngay
LOOP:  JB      PULSE, LOOP
    CLR     TF0
    SETB    TR0
    MOV     TH0,#00  ; nạp hệ số chia cho Timer0
    MOV     TL0,#00
WAITHI:  JNB     PULSE,WAITHI; chờ mức cao của xung
WAITPL:  JB      PULSE,WAITPL; chờ mức thấp của xung
    JB      TF0, LOOP  ; quay lai, nếu tràn
    MOV     A, TH0
    ACALL    TRANS ; phát byte cao
    MOV     A, TL0
    ACALL    TRANS; phát byte thấp
    MOV     A,  ','
    ACALL    TRANS; phát dấu ','
    SJMP    LOOP
TRANS:   JNB    TI, TRANS  ; CTC phát 1 byte
    CLR    TI              ; xoá cờ TI
    MOV    SBUF, A    ; phát byte trong Acc
    RET
    END
```

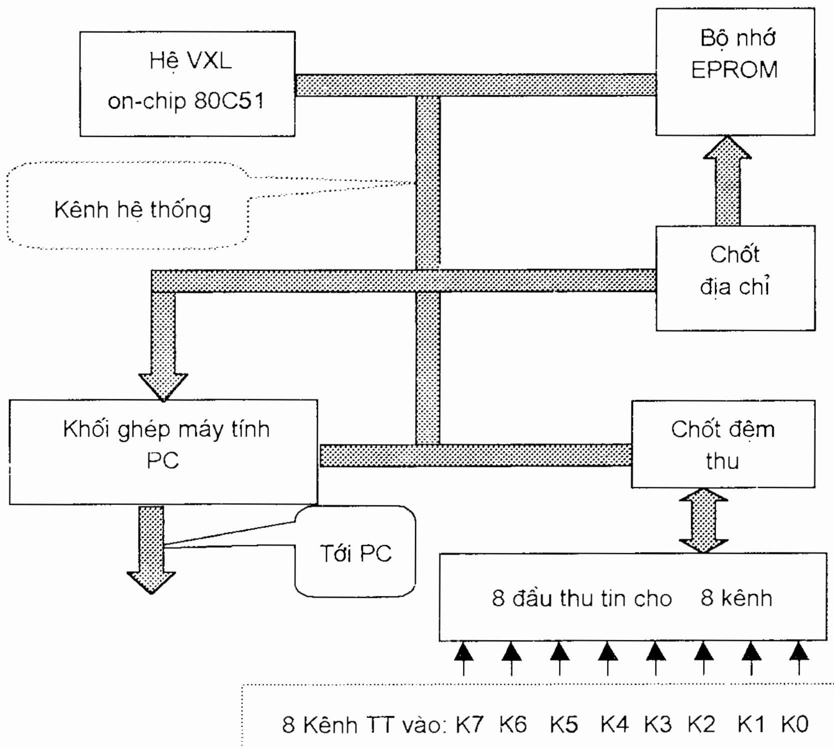
Hệ chức năng 3. Thiết kế hệ thu tin 8 kênh

Để dễ theo dõi, ta sử dụng lại yêu cầu thiết kế hệ thu tin đa kênh ở chương 4 với một vài thay đổi:

- Thông tin về trạng thái cờ tràn của các kênh sẽ được chuyển vào máy tính PC khi có yêu cầu (mã yêu cầu= 5Ah).
- Số lượng kênh là 8 chứ không phải là 6.
- Hệ không cần bàn phím và hệ hiển thị, tức là hệ đóng vai trò là hệ sơ cấp. Hệ thứ cấp dùng để xử lý đầy đủ là máy tính PC.

Thiết kế phần cứng của hệ thu tin 8 kênh

Căn cứ vào chức năng của hệ vi xử lý cần thiết kế, hệ thống thu tin 8 kênh có sơ đồ khối được lựa chọn như thể hiện trên hình 11.3.



Hình 11.3. **Sơ đồ khối của hệ thống thu tin 8 kênh**

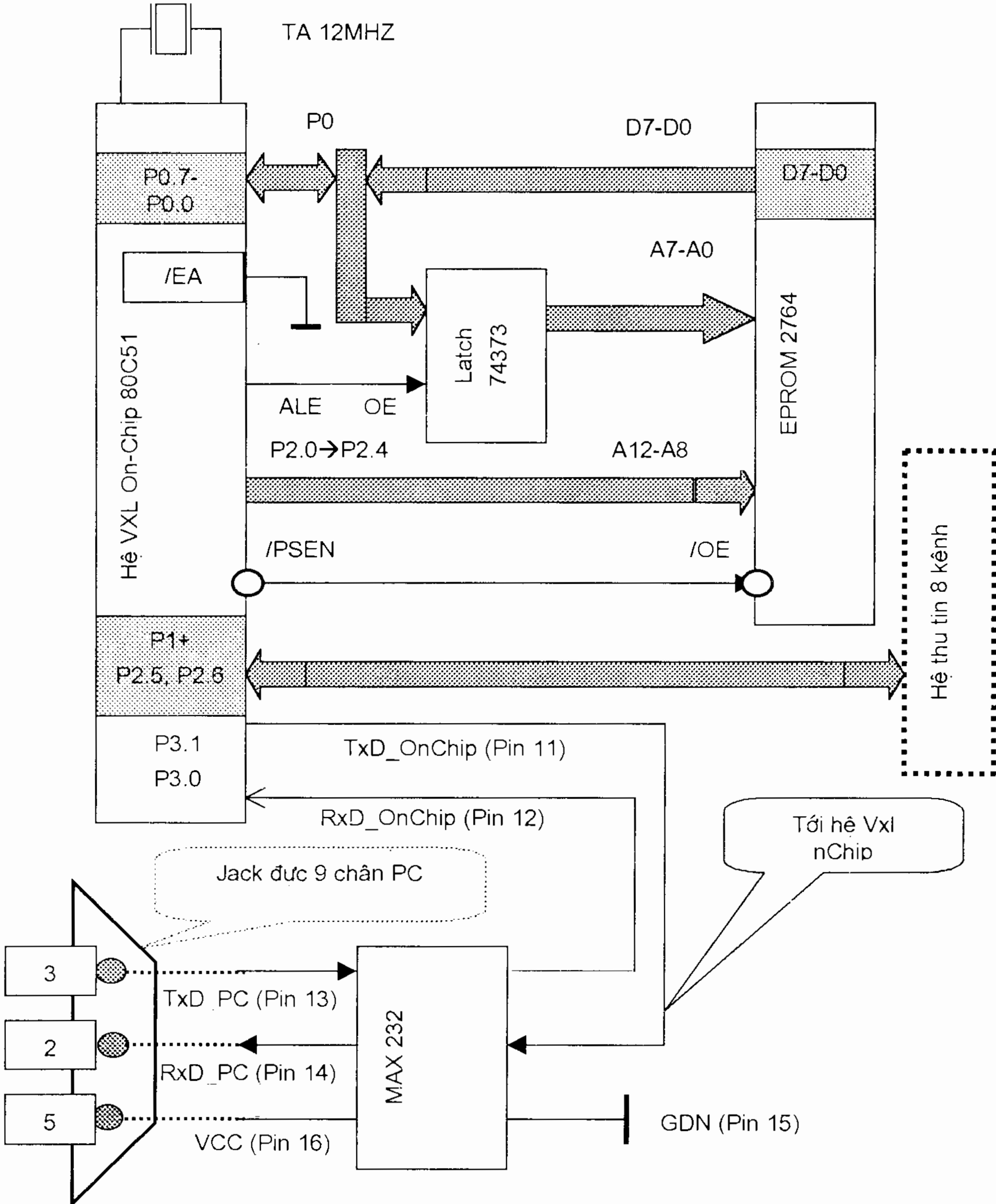
Chức năng của các thành phần

Hệ thống thu tin 8 kênh chọn hệ vi xử lý on-chip 80C51 làm hệ xử lý trung tâm. Tuy trong hệ vi xử lý on-chip có các bộ nhớ chương trình <EPROM> nội trú và bộ nhớ dữ liệu <RAM> nội trú, song để thuận tiện cho quá trình thiết kế hệ thống thu tin 8 kênh này, ta chọn phương án tổ chức không gian bộ nhớ cho hệ vi xử lý on-chip là sử dụng <RAM> nội trú và <EPROM> ngoại trú (dung lượng nhớ 8kb). Như vậy, việc nạp chương trình MONITOR cho hệ sẽ không phụ thuộc vào cấu trúc ROM của hệ vi xử lý on-chip.

Do sử dụng bộ nhớ chương trình có dung lượng nhớ 8kb ngoại trú nên để truy cập tới bộ nhớ này, hệ thống phải dùng chốt để chốt byte thấp của kênh địa chỉ.

Bộ thu tin 8 kênh thực hiện thu dạng tín hiệu là xung điện áp có thời điểm xuất hiện và thời gian tồn tại là đại lượng ngẫu nhiên, do đó để tin bị mất với xác suất thấp nhất thì ngoại vi thu tin được thiết kế theo phương pháp bẫy tín hiệu xung.

Tín hiệu sau khi được xử lý trong hệ vi xử lý on-chip sẽ được chuyển sang máy tính để thực hiện phép xử lý thứ cấp theo yêu cầu khi có mã 5Ah từ máy tính đưa sang.



Hình 11. 4. **Sơ đồ chức năng của hệ thu tin 8 kênh**

Hệ vi xử lý On-chip

Hệ xử lý trung tâm sử dụng hệ vi xử lý on-chip 80C51 hoạt động ở tần số *focs* = 12Mhz với bộ dao động bên trong và có thạch anh dao động 12Mhz, tụ ổn định C1 = C2 = 30μf.

Khởi tạo hệ vi xử lý on-chip bằng mạch <RESET> gồm tụ C3 = 10μf, R1 = 8,2kΩ, nối dương nguồn Vcc = +5volt qua tụ C3 vào chân <RST> của on-chip và qua R1 xuống mass. Nhờ mạch <RESET> này ta đảm bảo thời gian chân <RST> phải đặt ở mức +Vcc không vượt quá 1ms, mặt khác sẽ đảm bảo thời gian ổn định dao động không vượt quá 10ms sau khi cấp nguồn.

Khi khởi động lại thì trong các thanh ghi chức năng đặc biệt sẽ có giá trị 00h ngoại trừ các chốt cổng, con trỏ ngăn xếp và thanh ghi <SBUF>.Sau khi khởi tạo, các chốt cổng có giá trị là FFh, con trỏ ngăn xếp có giá trị là 07h và thanh ghi <SBUF>thì có giá trị không xác định. Riêng đối với RAM khi khởi tạo lại thì không bị tác động mà nội dung trong nó là ngẫu nhiên .

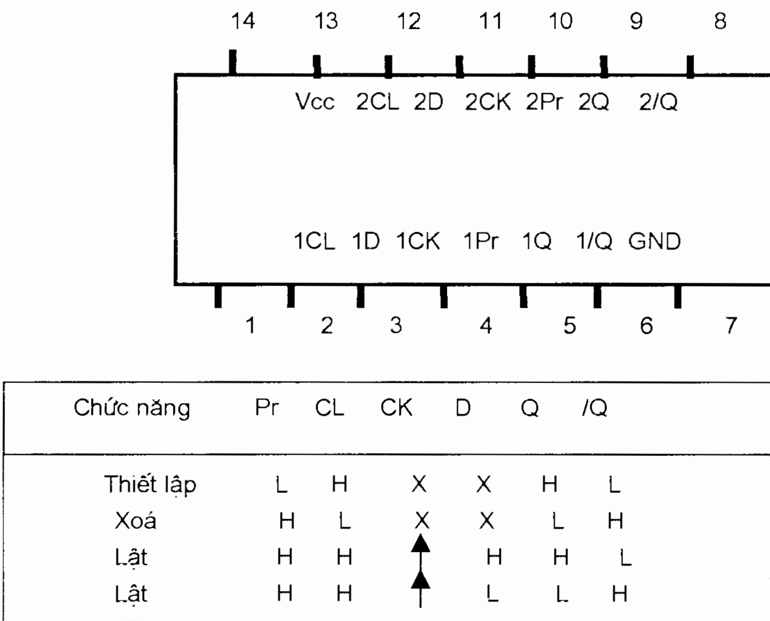
Tổ chức bộ nhớ của hệ thống thu tin

Trong hệ vi xử lý on-chip 80C51 có tổ chức cả 2 dạng bộ nhớ là bộ nhớ chương trình <EPROM> gồm 4 kb và bộ nhớ dữ liệu <RAM> gồm 128 byte đảm bảo cho hệ thống lưu trữ dữ liệu được phù hợp. Song với <EPROM> nội trú bên trong cần phải có thiết bị nạp chuyên dụng, mặt khác chương trình MONITOR khi cần thay đổi không được thuận tiện, cho lên hệ thống thu tin chọn phương án dùng bộ nhớ chương trình ngoại trú, do đó chân chọn bộ nhớ chương trình </EA> của hệ vi xử lý on-chip 80C51 được nối mass. Hệ thống sử dụng bộ nhớ dữ liệu nội trú tuy dung lượng bộ nhớ này nhỏ nhưng lượng thông tin của 8 kênh với <RAM>128byte vẫn đảm bảo thoả mãn chức năng.

Bô nhớ chương trình <EPROM> trong hệ thống là vi mạch loại 2764 có dung lượng nhớ là 8kb. Truy cập tới <EPROM> bằng địa chỉ 13 bits từ hệ vi xử lý on-chip 80C51 với 8 bits thấp qua cổng P0 và 5 bits cao qua cổng P2. Mã lệnh được đọc từ <EPROM> nhập vào hệ vi xử lý on-chip cũng qua cổng P0, do đó byte thấp của địa chỉ sẽ được chốt bằng 74373 để phân định giữa thời gian truy cập địa chỉ và thời gian đọc mã lệnh. Hệ vi xử lý On-chip điều khiển đọc<EPROM>bằng tín hiệu logic 0 phát qua chân </PSEN> vào chân </OE> của <EPROM>. Do có tín hiệu điều khiển đọc bộ nhớ chương trình riêng biệt với bộ nhớ dữ liệu, nên cho phép tổ chức bộ nhớ chương trình có địa chỉ song song với bộ nhớ dữ liệu (<EPROM> được định vị địa chỉ từ 00h đến 0FFFh) , vì vậy cấu trúc của hệ không cần phải có bộ giải mã địa chỉ bộ nhớ. Nối ghép <EPROM> trong hệ thống thu tin ngoài chân /OE và các chân địa chỉ, chân dữ liệu nối trực tiếp hoặc gián tiếp qua vi mạch chốt với hệ vi xử lý on-chip thì chân nguồn Vcc, Vpp và chân /PGM nối với +5vol, chân /CE nối mass.

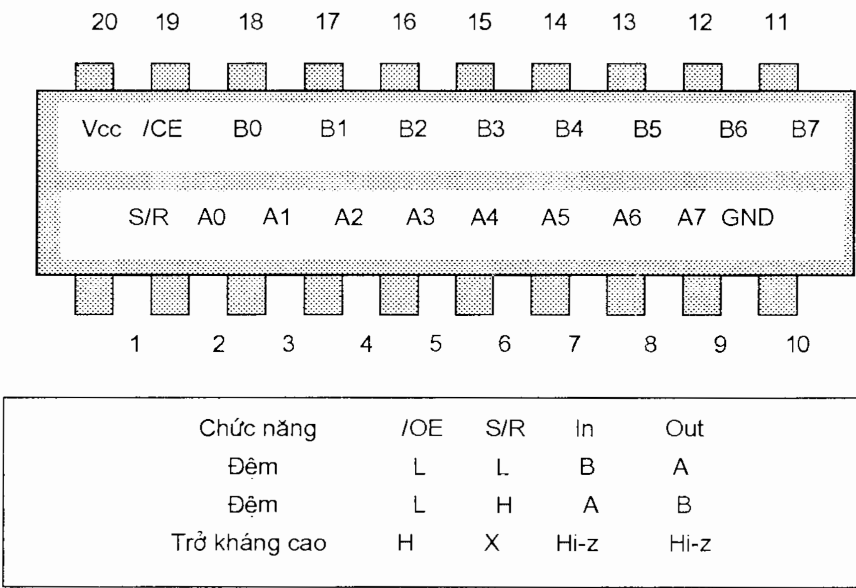
Ngoại vi thu tin

Ngoại vi thu tin 8 kênh sử dụng 8 flip-flop loại D <D-FF> làm đầu thu tin dạng bẫy xung được thực hiện trên 4 IC loại 74LS74, mỗi IC này gồm 2 flip-flop có sơ đồ chân và bảng trạng thái như hình 11.5.



Chức năng	Pr	CL	CK	D	Q	/Q
Thiết lập	L	H	X	X	H	L
Xoá	H	L	X	X	L	H
Lật	H	H	↑	H	H	L
Lật	H	H	↑	L	L	H

Hình 11.5. **Sơ đồ chân và bảng trạng thái của <D-FF>74LS74**



Chức năng	/OE	S/R	In	Out
Đệm	L	L	B	A
Đệm	L	H	A	B
Trở kháng cao	H	X	Hi-z	Hi-z

Hình 11.6. **Sơ đồ và bảng trạng thái của 74LS245**

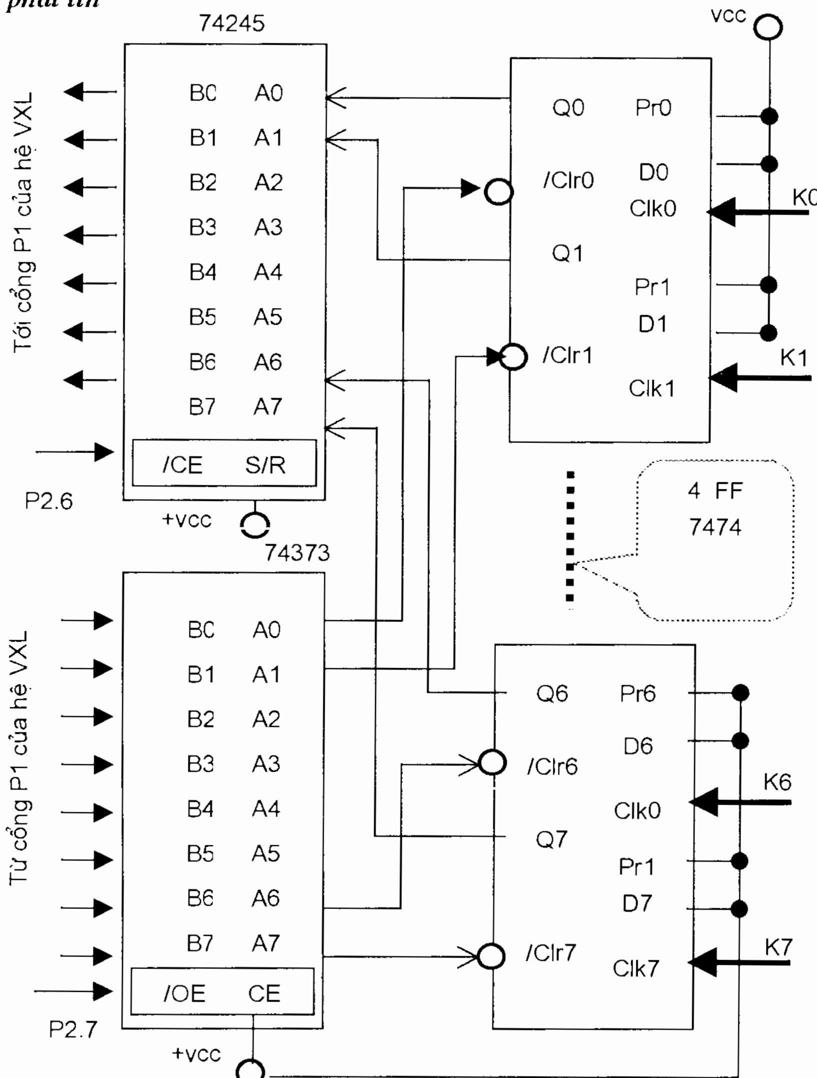
Nối ghép các flip-flop theo nguyên tắc ở dạng bẫy xung sao cho khi có tín hiệu xung ở đầu vào mỗi kênh thì flip-flop là đầu thu tương ứng của kênh được lật trạng thái từ mức logic 0 lên mức logic 1 để ghi nhận tin. Để thông tin ở mỗi kênh không

bị mất, thông tin của các kênh luôn được đọc vào on-chip, sau mỗi lần đọc thông tin, kênh nào có tin đã thu được thì flip-flop của kênh đó sẽ được xoá và thiết lập về trạng thái ban đầu để sẵn sàng nhận xung tín hiệu tiếp theo. Nối ghép các flip-flop làm đầu thu như hình 11.7.

Trong sơ đồ hình 11.7, đầu thu tin được nối với hệ vi xử lý on-chip 80C51 qua cổng P1. Chốt đệm đọc thu dữ liệu được chọn loại 74LS245 là vi mạch ba trạng thái, có sơ đồ chân và bảng trạng thái như hình 11.6, chốt đệm ghi xoá thiết lập được chọn loại 74LS373.

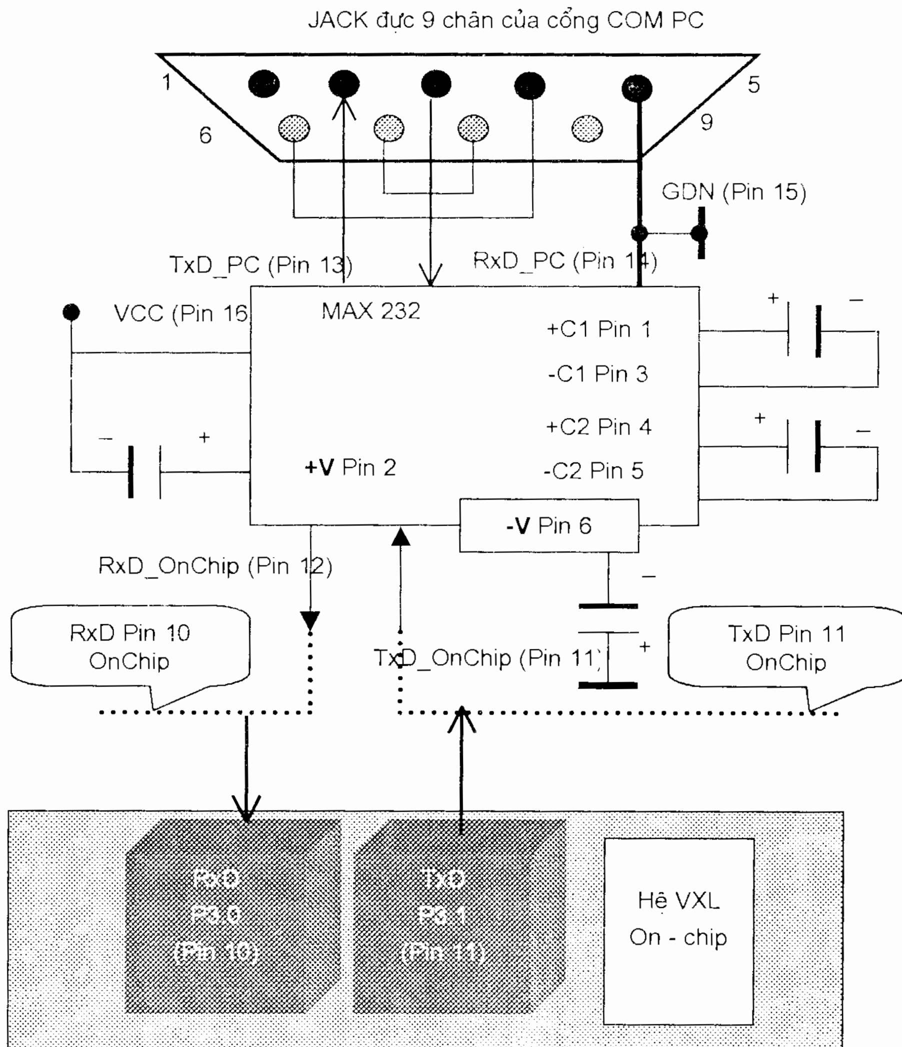
Khi thu tin, 8 bits từ đầu thu được đọc vào cổng P1 của hệ vi xử lý on-chip qua IC 74245 bởi tín hiệu P2.6, sau đó IC 74245 được chuyển sang trạng thái trở kháng cao để nhường <bus> cho chức năng Reset flip-flop. Khi lập trạng thái cho các đầu thu, byte mã 8 bit được truyền từ cổng P1 qua IC 74373 bởi tín hiệu P2.7. Lưu ý là chỉ những đầu thu có tin đã được đọc mới được lập lại trạng thái.

Mạch phát tin



Hình 11.7. Tổ chức ngoại vi thu tin 8 kênh

Để bắt tay chặt chẽ giữa hệ thống thu tin sơ cấp trên hệ vi xử lý on-chip với hệ xử lý tin thứ cấp trên máy tính IPM-PC, hệ sử dụng phương pháp truyền tin nối tiếp bằng UART qua cổng mạch tương thích TTL-RS 232 (IC MAX 232) và qua jack COM 9 chân của PC (*1-DCD, 2-RxD, 3- TxD, 4- DTR, 5- GND, 6- DSR, 7-RTS, 8- CTS, 9- RI*). Sơ đồ ghép được biểu diễn trên hình 11.8.



Hình 11.8. **Giao tiếp qua cổng COM với máy tính PC**

XÂY DỰNG PHẦN MỀM ĐIỀU KHIỂN CHO HỆ THU TIN 8 KÊNH

Chức năng của phần mềm xử lý sơ cấp

Do chức năng của hệ thống thực hiện thu tín hiệu ngẫu nhiên cho nên chương trình điều hành hệ thống luôn kiểm tra theo chu kỳ trạng thái của các flip-flop ở đầu vào, để xác định xem có tín hiệu xuất hiện ở các kênh hay không. Nếu kênh nào xuất hiện tín hiệu, thì lập tức tín hiệu được chuyển vào vùng nhớ tương ứng của kênh.

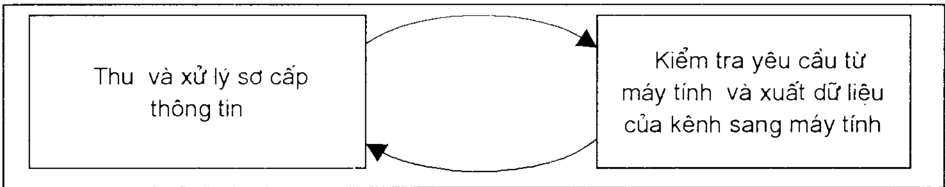
Thông tin thu được còn phải xử lý thứ cấp và hiển thị trên máy tính PC, do đó hệ thống cũng luôn kiểm tra theo chu kỳ yêu cầu từ máy tính.

Theo yêu cầu của máy tính, hệ thống xuất thông tin về trạng thái cờ tràn của các kênh sang máy tính.

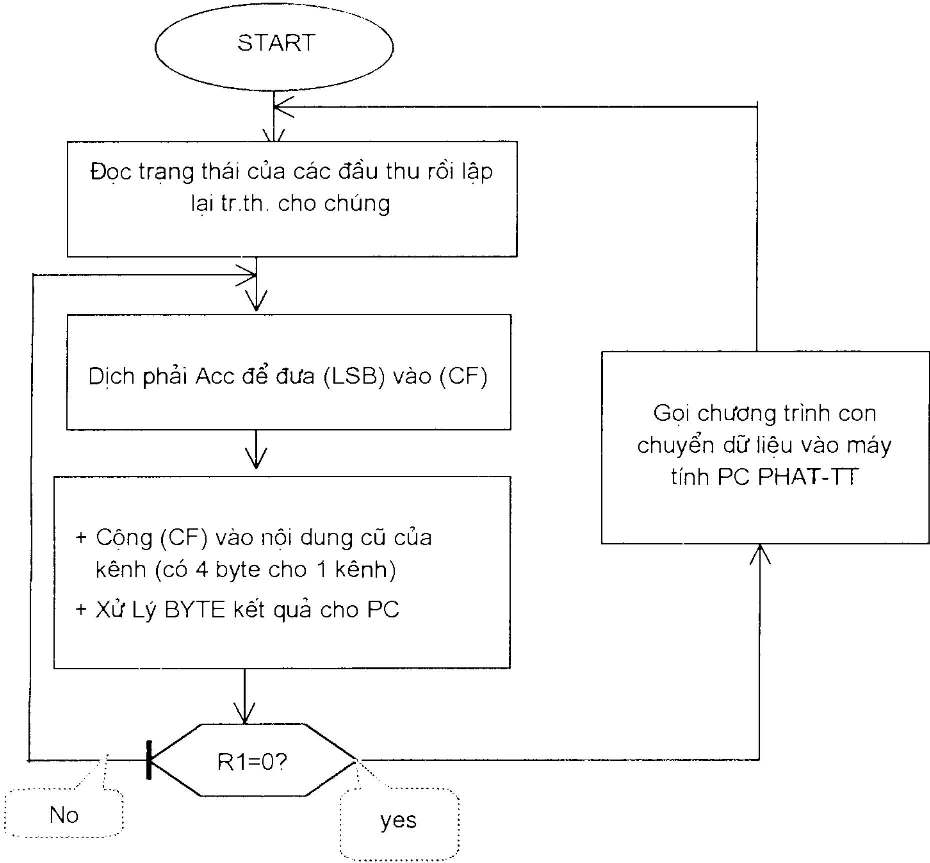
Để giảm tới mức thấp nhất sự mất mát thông tin ở đầu thu, chương trình điều hành phải dành ưu tiên cho chức năng kiểm tra trạng thái các flip-flop, do đó sau mỗi lần kiểm tra và đáp ứng yêu cầu từ máy tính thì chương trình sẽ quay lại ngay chức năng kiểm tra trạng thái các flip-flop.

Xây dựng thuật toán điều khiển

Từ những chức năng cơ bản của hệ thống, thuật toán của chương trình cho phần xử lý sơ cấp được thể hiện như hình 11.9.

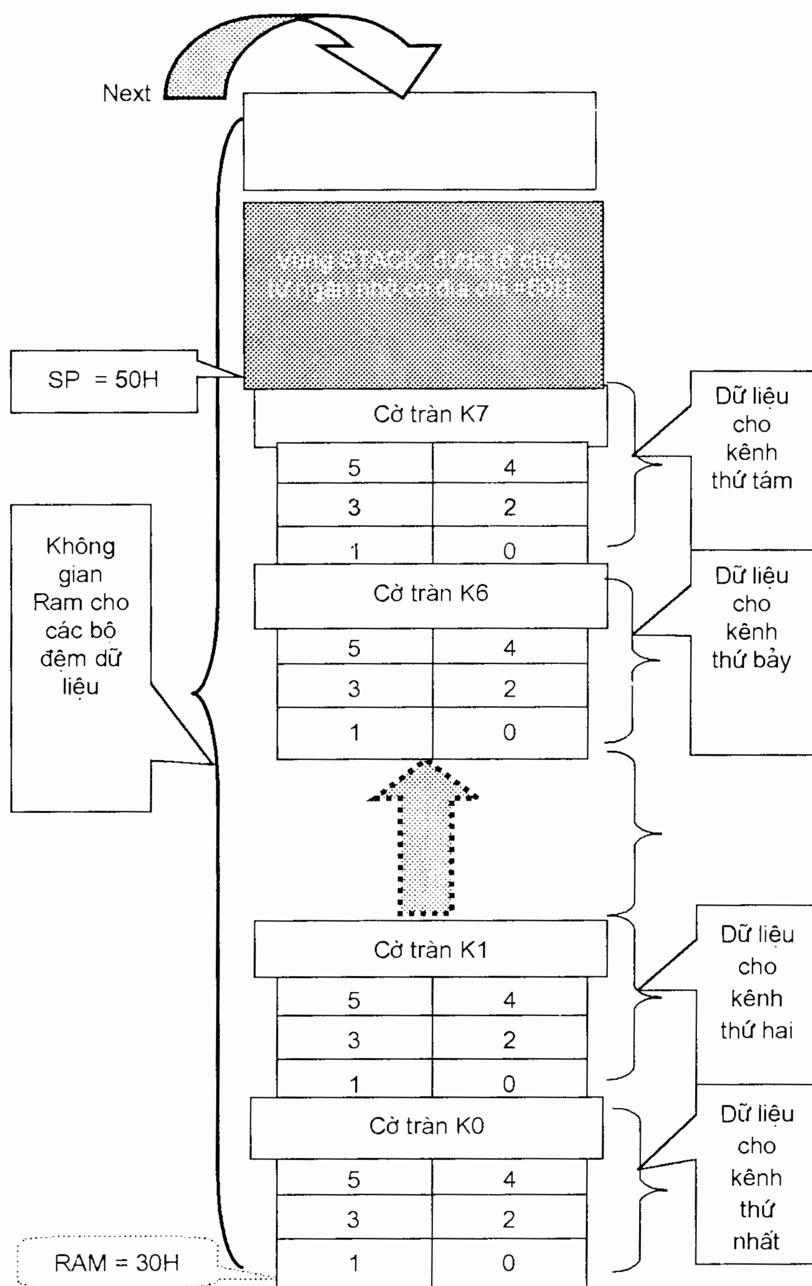


Hình 11.9. **Thuật toán điều khiển cho hệ thống thu tin**



Hình 11.10. **Lưu đồ thuật toán của chương trình con thu tin**

Tổ chức vùng đệm cho các kênh cũng thực hiện theo phương án của hệ thu tin trong chương 4 đã chỉ ra (hình 11.11).



Hình 11.11. **Tổ chức vùng đệm kênh cho hệ vi xử lý**

Để hệ thống thu tin, từ thuật toán của chương trình điều hành cho phép xây dựng lưu đồ thuật toán như hình 11.9. Theo lưu đồ thuật toán này chương trình điều hành phải thực hiện liên tiếp 2 chương trình con, đó là chương trình con thu xử lý sơ cấp thông tin <THU-TT> và chương trình con truyền thông tin sang máy tính <PHAT-TT>.

Dữ liệu sau khi thu và xử lý sơ cấp cần được lưu trữ vào trong bộ nhớ dữ liệu RAM. Trong phần thiết kế hệ thu tin trên, đã đề cập đến phương án chọn RAM nội trú bên trong hệ vi xử lý On-chip 80C51 để lưu trữ dữ liệu. Trong bộ nhớ RAM này vùng ô nhớ có địa chỉ từ 30h đến 7Fh được dành cho lưu trữ dữ liệu, vì vậy tổ chức vùng đệm cho các kênh trong RAM là vùng ô nhớ có địa chỉ từ 30h đến 4Fh để lưu trữ dữ liệu cho 8 kênh. Mỗi kênh sử dụng 4 byte trong đó 3 byte để lưu trữ số lượng xung của kênh và byte thứ 4 dùng để xác định số lượng xung tràn (byte tràn).

Vùng ô nhớ có địa chỉ từ 00h đến 1Fh của RAM trong on-chip dành riêng cho các băng thanh ghi. Hệ thống thu tin sẽ sử dụng các thanh ghi trong băng thanh ghi để làm con trỏ kênh và sử dụng cho các mục đích khác. Khi được RESET lại thì con trỏ ngăn xếp <SP> chỉ về địa chỉ 07h, do đó chương trình phải đặt con trỏ ngăn xếp lên cao hơn để có thể sử dụng vùng RAM thấp, cụ thể chương trình chọn địa chỉ 50h để làm giá trị khởi đầu cho ngăn xếp <STACK> và cho phép nó phát triển lên phía trên từ địa chỉ này. Tổ chức không gian bộ nhớ dữ liệu được thể hiện như trên hình 11.11.

Chương trình nguồn xử lý sơ cấp

Chương trình con thu và xử lý thông tin được đặt tên là THU-TT được viết trên ngôn ngữ ASSEMBLY có dạng như sau:

```
          FLOP_IN        EQU  P2.6
          FLOP _OUT      EQU  P2.7
          Ram_Buf        EQU  30h
          LoopCount      EQU  R1
          Byte_Trans     EQU  R7
          PcCommand      EQU  5Ah
ORG    0000h            ; Địa chỉ đầu của chương trình
MOV  SP,#50h            ; Nạp con trỏ STACK.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;    Khởi đầu cho hệ vxl Onchip
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
RESET:
    MOV   TMOD,#20H   ; gate1=0, C/T1=0, mode cho Timer1= mode2 -
                      ;tự nạp lại  hệ số chia;
    MOV     TH1,#0FDH  ; tốc độ = 9600 baud
    MOV     TCON,#00H
    MOV   SCON,#52H;  Mode1 cho UART, máy phát (bit N1- cờ TI= 1)
                     ; sẵn sàng, cho phép thu (bit N4- cờ REN= 1).
    SETB     TR1       ; cho Timer1 chạy ngay để tạo tốc độ baud
```

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
;          Làm sạch RAM
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

MOV  R0,#00h        ; Nạp địa chỉ đầu tiên của RAM
MOV  R1,#7Fh        ; Nạp địa chỉ cuối cùng của RAM
LOOP1 : MOV  @R0, #00h    ; Đặt các ô nhớ của RAM về 0
        INC    R0               ; Tăng R0 lên 1.
        DJNZ   R1, LOOP1    ; Thoát khỏi vòng lặp nếu R1=0

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
" Thủ tục chuẩn bị thu tin "
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

        CLR  A              ; Xoá Acc <Acc=00h>
        CPL   A             ; Lấy bù Acc<Acc=FFh>
        MOV  P1,A          ; chuyển từ Acc ra cổng P1
        CLR FLOP _OUT
        SETB FLOP s_OUT  ; lập trạng thái sẵn sàng thu tin của flip-flop.
        SETB FLOP _IN       ; khoá cổng vào

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
"Thủ tục thực hiện chương trình chính "
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

LOOP :  ACALL THU_TT      ;  Gọi chương trình con THU-TT
        ACALL PHAT_TT    ;  Gọi chương trình con PHAT-TT
        AJMP    LOOP          ;  Trở lại vòng lặp
            ----------------- --------------------------------
```

Đầu tiên lệnh giả "ORG" chỉ cho chương trình dịch biết địa chỉ đầu tiên của chương trình nguồn và định địa chỉ của ngăn xếp trong RAM bằng lệnh nạp giá trị <# 50h> vào con trỏ ngăn xếp.

Do khi hệ được RESET thì nội dung trong RAM là ngẫu nhiên, cho nên chương trình phần mềm phải thực hiện thủ tục làm sạch RAM trước khi sử dụng RAM làm vùng đệm lưu trữ dữ liệu.

Sau khi khai báo địa chỉ của các chốt đệm mạch thu phát tin hệ thực hiện chương trình chính bằng vòng lặp LOOP để thu phát tin.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
"Thủ tục thực hiện chương trình con THU-TT "
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

THU_TT: CLR FLOP _IN
        MOV  A,P1      ; đọc trạng thái flip-flop qua 74245
        SETB FLOP _IN
        MOV  B, A        ; chuyển data từ Acc vào thanh ghi cơ sở B.
        CPL   A          ; Lấy bù Acc
        MOV  P1,A        ;  chuyển từ Acc ra cổng P1.
        CLR FLOP _OUT
        MOV  A,P1      ; lập trạng thái flip-flop
        SETB FLOP _OUT
        MOV    LoopCount, #08h   ;  Nạp số vòng lặp vào R1
        MOV    R0, Ram_Buf ;=30h  Nạp địa chỉ vùng nhớ đầu.
        MOV BYTE_TRANS, #00h  ; xoá byte_trans cho PC
LOOP2:  MOV    A,B           ;  lấy lại B và → A
        RRC    A             ;  Quay phải  Acc qua cờ nhớ CF
        MOV    B,A           ;  Chuyển A vào B
;------------------------------------------------------
; xử lý byte thứ nhất của kênh
;------------------------------------------------------
        CLR      A           ; Xoá Acc
        ADDC   A,@R0        ; Cộng CF vào byte được trỏ tới
        DA A                ; Chỉnh thập phân byte kết quả
        MOV   @R0, A        ; Chuyển từ A vào RAM
        INC   R0            ; Tăng lên byte thứ 2 của kênh
;------------------------------------------------------
; xử lý byte thứ hai của kênh
;------------------------------------------------------
        CLR      A           ;  Xoá Acc.
        ADDC   A,@R0        ;  Cộng CF vào byte được trỏ tới
        DA A                ; Chỉnh thập phân byte kết quả
        MOV   @R0,A         ; Chuyển từ A vào RAM
        INC   R0            ; Tăng lên byte thứ 3 của kênh
```

```
;-----------------------------------------------
; xử lý byte thứ ba của kênh
;-----------------------------------------------
            CLR      A            ; Xoá Acc.
            ADDC   A,@R0     ; Cộng CF vào byte được trỏ tới
            DA A                   ; chỉnh thập phân byte kết quả
            MOV   @R0, A      ; Chuyển từ A vào RAM
            INC   R0                ; Tăng lên→ byte tràn của kênh.
;-----------------------------------------------
; xử  lý byte cờ tràn của kênh
;-----------------------------------------------
            CLR      A            ; Xoá Acc.
            RLC   A               ; Quay trái Acc qua cờ CF
           ORL A, @R0   ;cộng logic với giá trị cũ
            MOV   @R0, A     ; Chuyển A vào RAM
           INC   R0              ;Tăng lên→ byte thứ nhất của kênh tiếp theo .
;------------------------------
; xử  lý byte_Trans cho PC
;------------------------------
           RRC A; chuyển Acc0→CY
           MOV  A,        Byte_Trans
           RRC   A    ; Quay phải Acc qua cờ CF để đưa CY vào bên trái,
                          ; còn các bit cũ của byte_Trans → phải
           MOV Byte_Trans, A
;------------------------------------------------------------------------
; kiểm tra số kênh đã xử lý : chưa hết thì quay về làm tiếp
;------------------------------------------------------------------------
        DJNZ   LoopCount, LOOP2   ; thoát khỏi vòng lặp 2 nếu R1=0
        RET    ; Kết thúc chương trình con THU-TT quay về chương trình chính
 ------------------------------------------------------------------
```

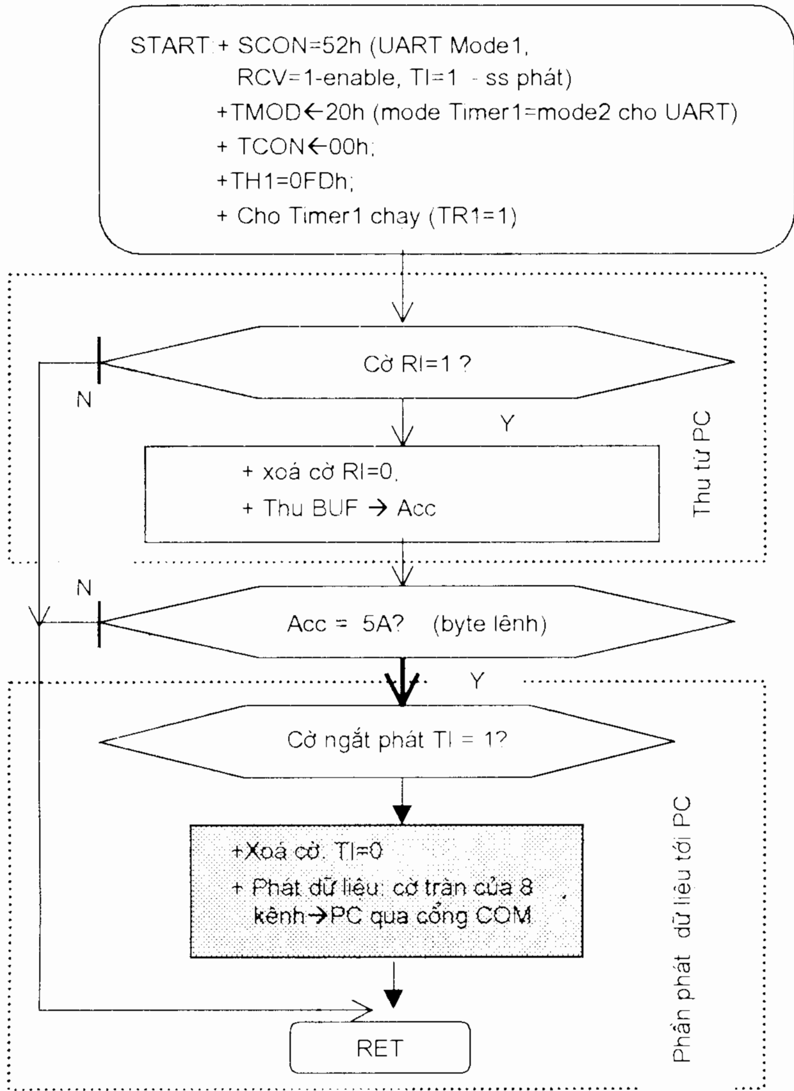
Nhiệm vụ của chương trình con "THU_TT " là đọc trạng thái các flip-flop của các kênh, lưu trữ thông tin vào vùng nhớ tương ứng của kênh.

Sau mỗi lần hệ đọc nội dung các đầu thu xong, hệ tự động chuyển sang xử lý dữ liệu trong hệ vi xử lý on-chip bằng vòng lặp LOOP2 với số vòng lặp bằng 8 được xác định trong thanh ghi R1 của băng thanh ghi thứ nhất.

Trong mỗi vòng lặp của LOOP2 phải thực hiện lệnh RRC A để quay phải thanh ghi tích luỹ qua cờ nhớ <CF> một bit, sau đó thực hiện cộng giá trị của kênh trong RAM với cờ nhớ và thực hiện lệnh DJNZ R1, LOOP2 để giảm giá trị trong LoopCount-R1 đi một đơn vị sao cho sau mỗi lần giảm 1 đơn vị nếu giá trị còn lại trong LoopCount-R1 khác 0 thì vòng lặp LOOP2 lại được thực hiện, nếu giá trị còn lại trong LoopCount1 bằng 0 thì kết thúc vòng lặp LOOP2 và hệ thực hiện lệnh tiếp theo là lệnh gọi chương trình con PHAT-TT. Lưu ý rằng nhóm lệnh xử lý byte_trans đã đặt các bit cờ tràn của từng kênh vào đúng vị trí của mình (bit 0 là cờ tràn của kênh đầu tiên, bit 1 là cờ tràn của kênh thứ hai, cứ như vậy bit 7 là cờ tràn của kênh cuối cùng).

Chương trình chuyển thông tin cho PC thực chất là chuyển một byte mang thông tin về trạng thái cờ tràn của các kênh khi PC có lệnh là mã 5Ah đưa sang hệ xử lý sơ cấp là hệ vi xử lý onc-hip.

Căn cứ yêu cầu của modul chức năng này thì thuật toán của chương trình con có dạng:

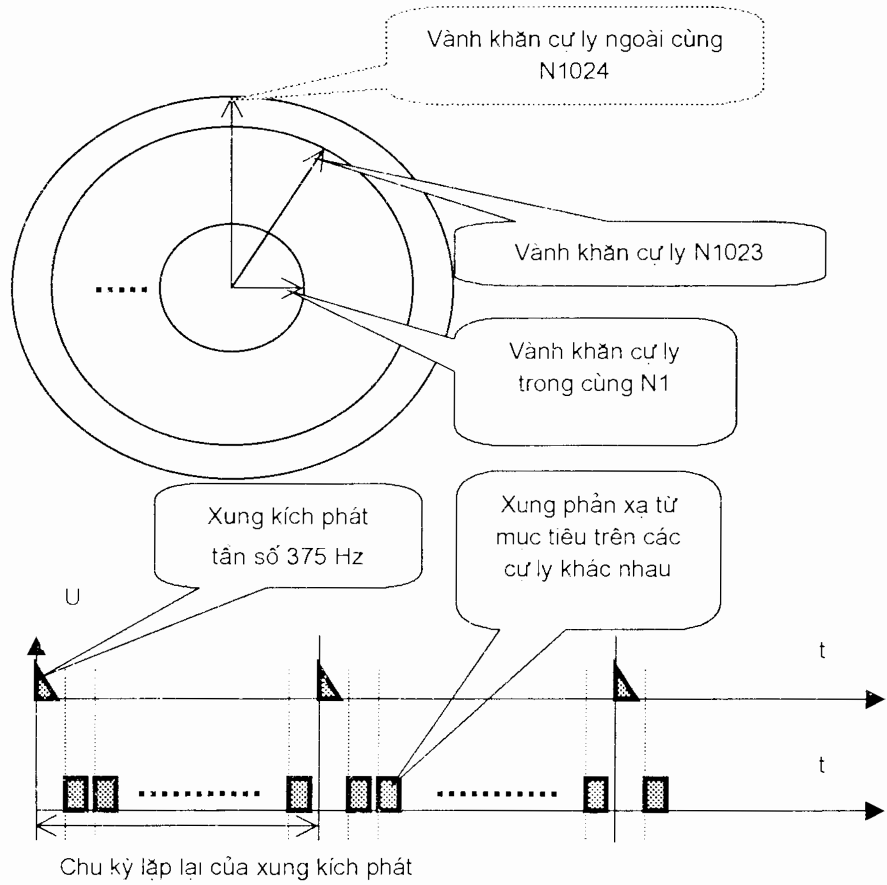


Căn cứ vào lưu đồ thuật toán đã xây dựng, chương trình nguồn PHAT_TT có dạng:

```
PHAT_TT: JNB   RI, END_ ;kiểm tra cờ RI=1?
        CLR RI                  ;xoá cờ RI=0
        MOV A, SBUF   ; thu từ PC qua UART
        CJNE A, PcCommand, END_  ; so sánh với 5Ah, sai→kết thúc
LOOP_TRANS: JNB TI, LOOP_TRANS  ; đúng thì phát cho PC
        CLR TI                  ;xoá cờ TI
        MOV A, BYTE_TRANS
        MOV SBUF, A ; chuyển nội dung byte_trans→PC
        END_:RET
```

Hệ chức năng 4. Thiết kế hệ phát hiện mục tiêu trên không

Mô tả chức năng hệ cần thiết kế



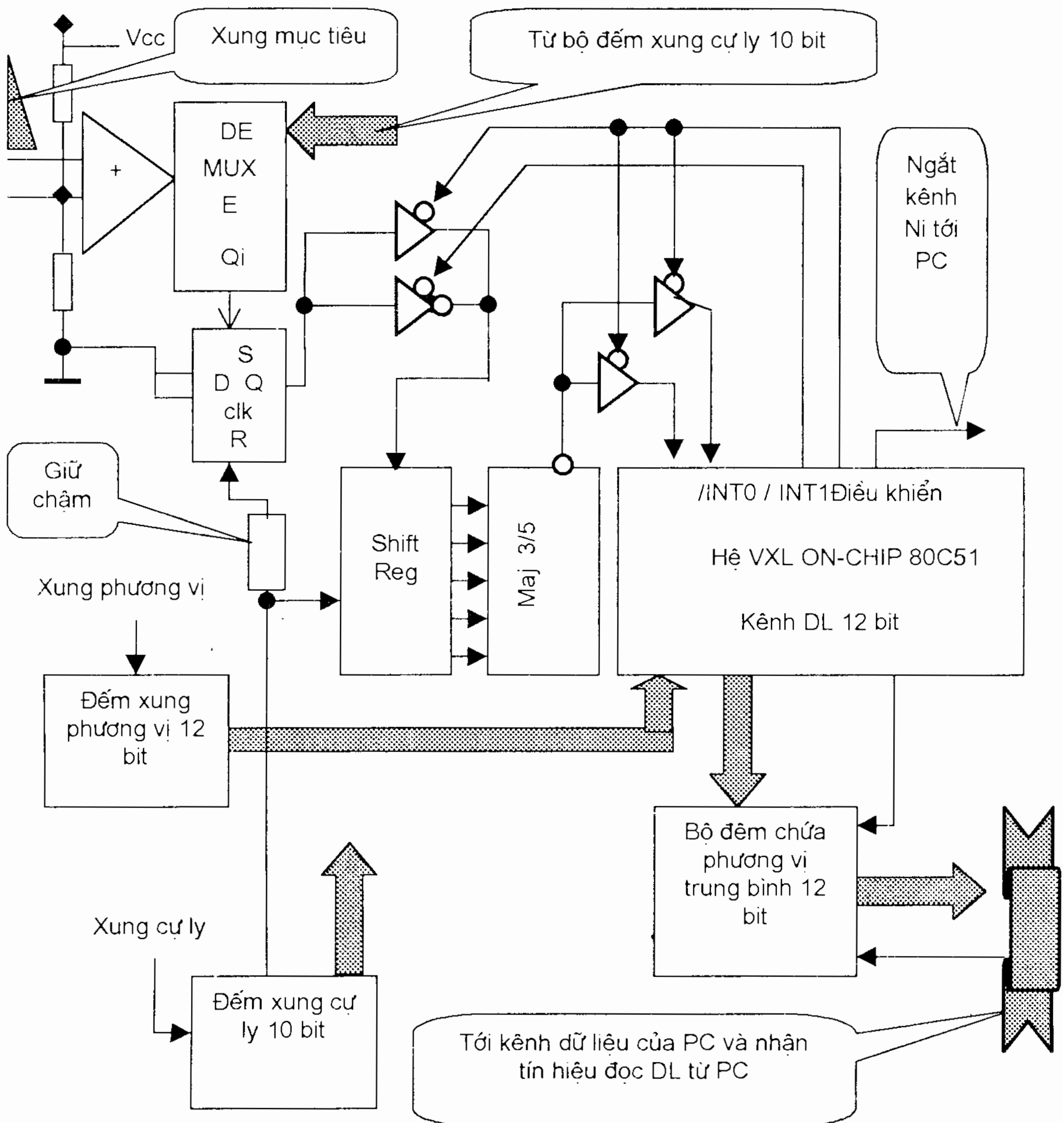
Hình 11.12. **Cự ly quan sát của hệ**

Cho hệ quan sát mục tiêu bằng phương pháp bức xạ xung kích phát trên anten quay vòng có tần số 375 Hz. Tín hiệu thu về là các xung phản xạ từ mục tiêu.Trong chu kỳ lặp lại của xung kích phát được chia thành 1024 khoảng thời

gian bằng nhau tương ứng với số lượng vành khăn cự ly (hình 11.12). Cự ly cực đại, ứng với vành khăn N1024 là 307200 m. Khi cánh sóng anten quét qua mục tiêu có thể thu được vài chục xung phản xạ. Nếu trong 5 lần thu mà có từ 3 xung phản xạ trở lên thì khẳng định mục tiêu đã xuất hiện. Tiếp theo, nếu trong 5 lần thu mà có từ 3 lần trở lên không có xung phản xạ thì khẳng định là cánh sóng đã đi qua mục tiêu. Mỗi lần phát hiện được mục tiêu phải cung cấp cho trung tâm xử lý số liệu về cự ly và phương vị trung bình của mục tiêu, biết rằng đĩa đo phương vị (cho 360 độ) phải chia thành 4096 phần bằng nhau.

Thiết kế phần cứng của hệ quan sát mục tiêu

Căn cứ vào chức năng của hệ cần thiết kế được mô tả ở phần trên, hệ thống quan sát mục tiêu phải có sơ đồ khối được lựa chọn như thể hiện trên hình 11.13 (cho kênh thứ i).



Hình 11.13. **Sơ đồ khối hệ quan sát mục tiêu**

Tín hiệu thu về dưới dạng xung có biên độ không cố định được bộ so sánh ngưỡng biến đổi thành tín hiệu TTL đưa tới đầu E (Enable) của bộ phân kênh DEMUX 10 bit (tạo 1024 kênh phân biệt). Mỗi đầu ra Qi của DEMUX được đưa tới đầu vào SET của Flip-Flop để chốt tín hiệu nếu nó tồn tại. Đầu ra của Flip-Flop được đưa vào bộ ghi dịch 5 bit SHIFT REG qua tuyến cực tính không đảo đưa tới bộ tạo hàm đa số MAJ 3/5 nhằm phát hiện sự xuất hiện của mục tiêu theo tiêu chuẩn 3/5. Để xử lý tình huống này, chương trình con phục vụ ngắt cài đặt trong hệ vi xử lý ON-CHIP sẽ được thực hiện khi có tín hiệu yêu cầu ngắt tới chân IT1. Chương trình con phục vụ ngắt IT1 sẽ đọc vào giá trị phương vị từ bộ đếm xung phương vị để đánh dấu vị trí đầu tiên của mục tiêu. Đồng thời chương trình con này sẽ điều khiển để đảo tuyến thông tin sao cho lần này chỉ tuyến đảo cực tính tín hiệu được đưa vào hoạt động. Điều này có nghĩa là hệ muốn phát hiện sự ra khỏi góc quan sát của mục tiêu cũng theo tiêu chuẩn 3/5 nhưng với cực tính ngược lại. Để xử lý tình huống này, chương trình con phục vụ ngắt cài đặt trong hệ vi xử lý ON-CHIP sẽ được thực hiện khi có tín hiệu yêu cầu ngắt tới chân IT0. Chương trình con phục vụ ngắt IT0 sẽ đọc vào giá trị phương vị từ bộ đếm xung phương vị để đánh dấu vị trí cuối của mục tiêu. Đồng thời chương trình con này sẽ tính toán giá trị trung bình của phương vị để chốt vào bộ đệm phương vị nhằm cung cấp cho máy tính theo yêu cầu.

Phương vị trung bình của mục tiêu được tính theo hệ thức:

PVtr.b = (PV đầu + PV cuối) / 2

Cự ly của mục tiêu được tính theo hệ thức:

i * (307200 m/1024 kênh) với i là số hiệu kênh

Điều này nghĩa là hệ chỉ cần cung cấp giá trị phương vị trung bình PVtr.b còn cự ly sẽ được máy tính tính ra khi nhận dạng vector ngắt Ni tới từ hệ vi xử lý ON-CHIP của kênh thứ i.

Đầu ra của bộ đếm xung cự ly 10 bit dùng để báo kết thúc một chu kỳ của xung kích phát. Tín hiệu kết thúc này sẽ dùng để dịch một bit trong thanh ghi dịch và xoá thông tin cũ trong Flip-Flop để Flip-Flop sẵn sàng nhận tín hiệu ở chu kỳ tiếp theo.

Đầu ra của bộ đếm xung cự ly 10 bit dùng cho bộ giải mã địa chỉ DEMUX nhằm phân phối thời gian cho 1024 kênh. Khoảng thời gian tính cho mỗi kênh được tính theo công thức:

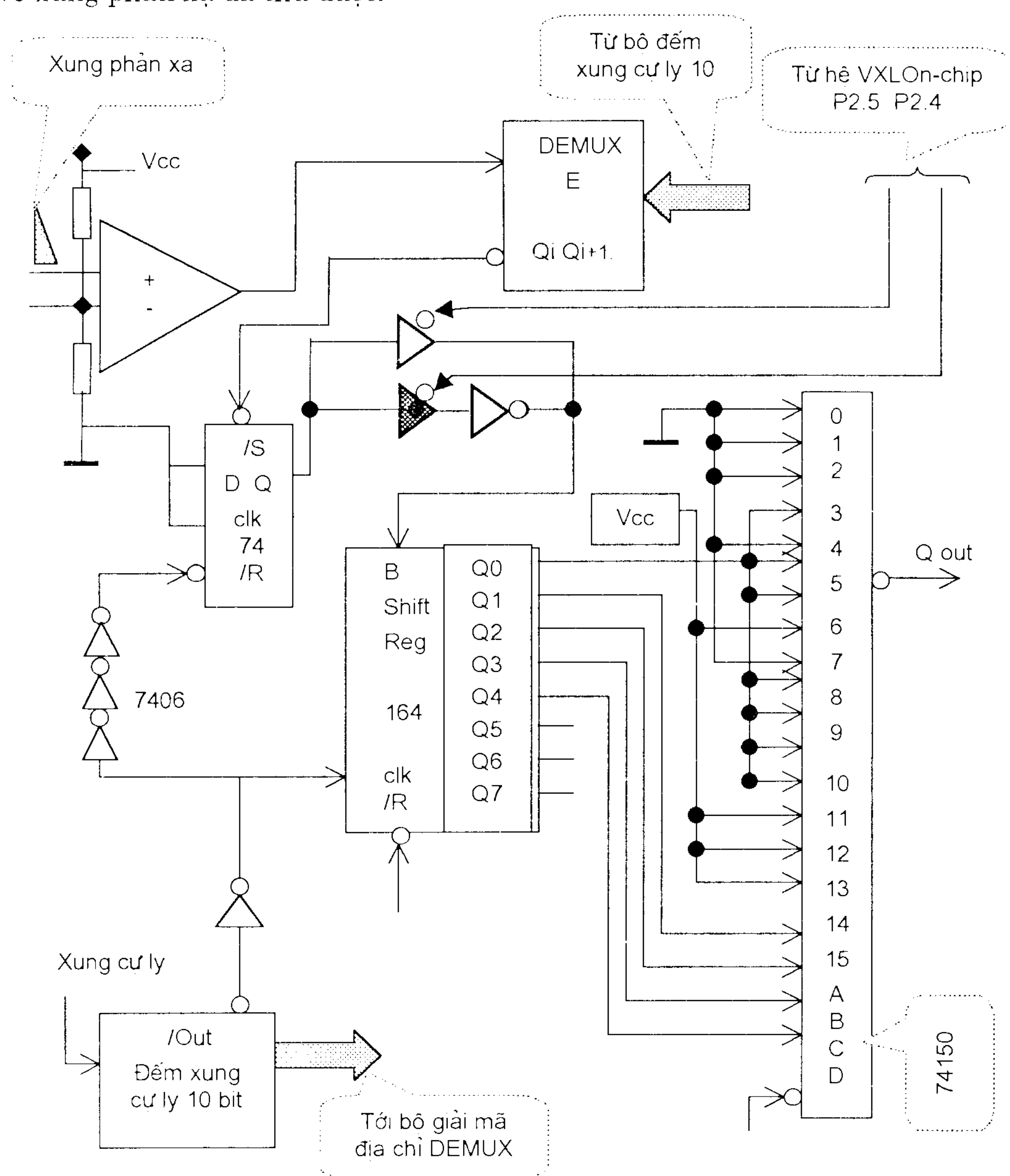
Tk = 1/ (375 Hz*1024) [sec]

Bây giờ tổ chức phần cứng chi tiết hơn cho từng thành phần chức năng.

Khối nhận xung phản xạ và xử lý thuật toán 3/5

Khối nhận xung phản xạ được tính từ đầu vào bộ so sánh ngưỡng tới đầu ra của bộ tạo hàm đa số MAJ. Khi tín hiệu phản xạ xuất hiện trên đầu + của bộ so

sánh ngưỡng được thiết kế từ bộ khuyếch đại thuật toán. Nếu tín hiệu vượt ngưỡng thì đầu ra sẽ xuất hiện mức logic cao. Mức logic này sẽ thiết lập trạng thái logic 1 cho đầu Qi của Flip-Flop (qua bộ DEMUX 10 bit). Giá trị 1 của Flip-Flop chính là tin về xung phản xạ đã thu được.



Hình 11.14. Khối nhận xung phản xạ và xử lý thuật toán 3/5

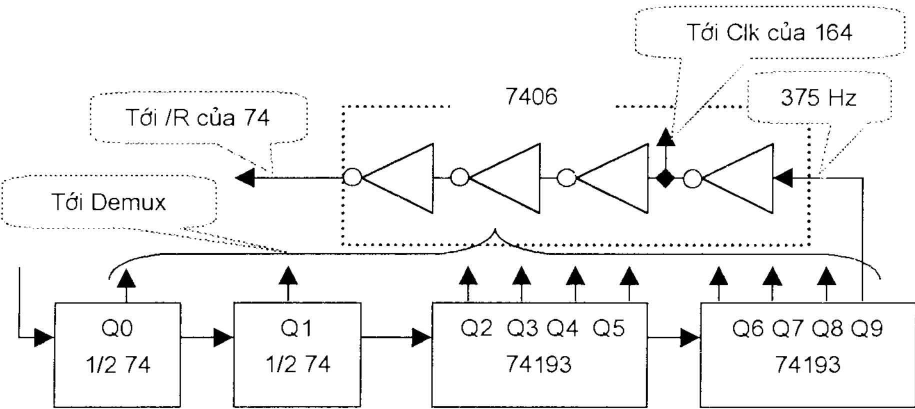
Tín hiệu từ đầu ra của Flip-Flop qua tuyến thuận (nhánh phía trên) tới đầu vào B của thanh ghi dịch 8 bit 164. Gõ nhịp dịch phải cho thanh ghi dịch được lấy từ đầu ra cuối của bộ đếm xung cự ly sau khi qua mạch NOT để bảo đảm cực tính đúng. 5 bit D0D1D2D3D4 được đưa vào mạch MUX 150 để tạo hàm đa số 3/5 theo

yêu cầu của thuật toán. Việc nối các chân tuân thủ theo bảng chức năng của hàm đa số Majority (bảng 11.3).

Bảng 11.3

D = Q4	C = Q3	B = Q2	A = Q1	Q0	Q out
0	0	0	0	0/1	Đầu vào 0=0/0 → Đầu vào 0 nối logic 0
0	0	0	1	0/1	Đầu vào 1=0/0→ Đầu vào 1 nối logic 0
0	0	1	0	0/1	Đầu vào 2=0/0→ Đầu vào 2 nối logic 0
0	0	1	1	0/1	Đầu vào 3=0/1→ Đầu vào 3 nối Q0
0	1	0	0	0/1	Đầu vào 4=0/0→ Đầu vào 4 nối logic 0
0	1	0	1	0/1	Đầu vào 5=0/1→ Đầu vào 5 nối Q0
0	1	1	0	0/1	Đầu vào 6=0/1→ Đầu vào 6 nối Q0
0	1	1	1	0/1	Đầu vào 7=1/1→ Đầu vào 7 nối logic 1
1	0	0	0	0/1	Đầu vào 8=0/0→ Đầu vào 8 nối logic 0
1	0	0	1	0/1	Đầu vào 9=0/1→ Đầu vào 9 nối Q0
1	0	1	0	0/1	Đầu vào 10=0/1→ Đầu vào 10 nối Q0
1	0	1	1	0/1	Đầu vào 11=1/1→ Đầu vào 11 nối logic 1
1	1	0	0	0/1	Đầu vào 12=0/1→ Đầu vào 12 nối Q0
1	1	0	1	0/1	Đầu vào 13=1/1→ Đầu vào 13 nối logic 1
1	1	1	0	0/1	Đầu vào 14=1/1→ Đầu vào 14 nối logic 1
1	1	1	1	0/1	Đầu vào 15=1/1→ Đầu vào 15 nối logic 1

Bộ đếm xung cự ly 10 bit được thiết kế trên cơ sở một chip 7474 ghép kiểu bộ đếm 2 bit. Bộ đếm này mắc nối tiếp với hai chip đếm 4 bit 74193 (chúng cũng được ghép với nhau) để tạo thành bộ đếm 10 bit (hình 11.15).

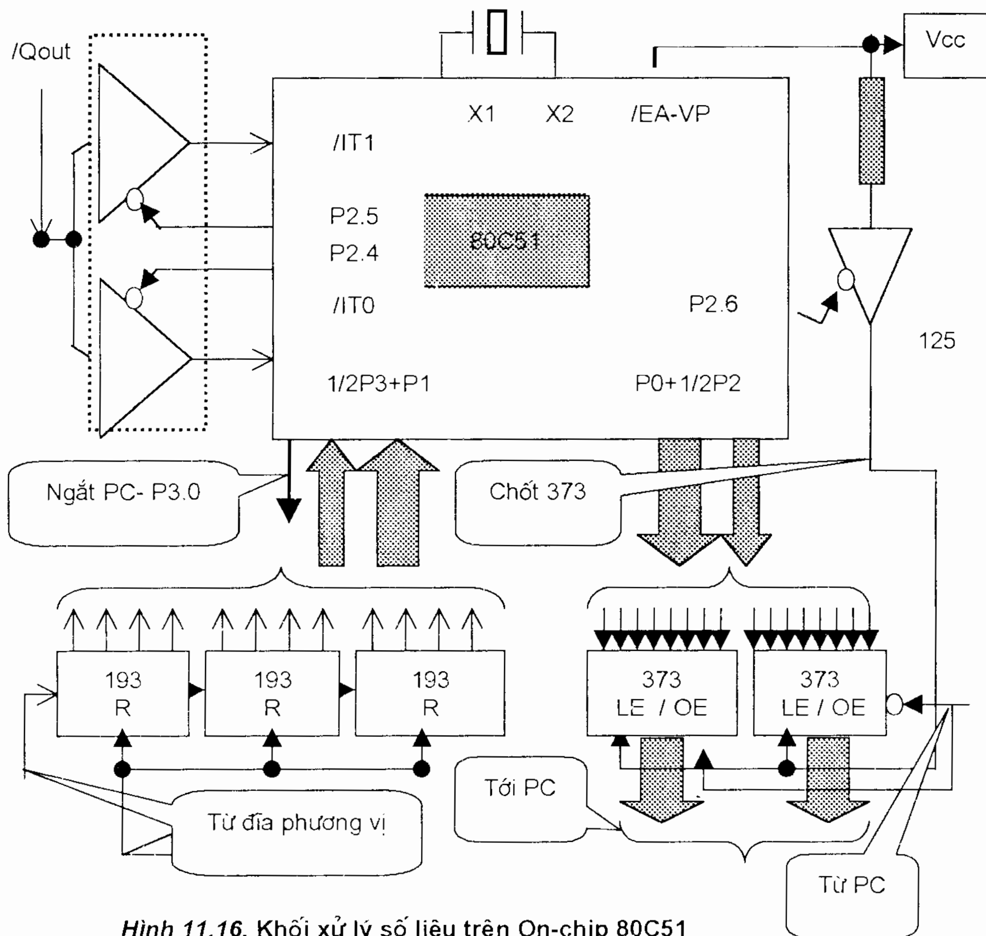


Hình 11.15. **Bộ đếm xung cự ly 10 bit**

Đầu ra Q9 có tần số 375 Hz chính là tần số lặp lại của xung kích phát. Tín hiệu này dùng để làm nhịp dịch phải cho thanh ghi dịch 164 (qua một mạch NOT là 1/4 chip 7406). Tín hiệu này cũng dùng để làm tín hiệu xoá Flip-Flop để đặt Flip-Flop vào trạng thái sẵn sàng làm việc sau khi đã qua bộ giữ chậm so với tín hiệu nhịp dịch phải cho thanh ghi dịch 164 bằng 3/4chip 7406.

Khối xử lý số liệu

Khối xử lý số liệu được xây dựng trên hệ vi xử lý On-chip 80C51 vì nó có các thành phần cần thiết cho xử lý theo chương trình và cho giao tiếp với máy tính. Các chương trình con xử lý số liệu được cài đặt bên trong ROM nội trú sẽ làm việc khi có ngắt tác động. Ngắt qua đầu IT1 sẽ xử lý tín hiệu cực tính dương nhằm phát hiện sự xuất hiện mục tiêu quan sát, còn ngắt qua đầu IT0 sẽ xử lý tín hiệu cực tính âm nhằm phát hiện sự kết thúc của mục tiêu quan sát. Hai tuyến dẫn tín hiệu này luôn được hệ vi xử lý On-chip điều khiển đảo pha nhau, bảo đảm tại một thời điểm chỉ có một tuyến được hoạt động.



Hình 11.16. Khối xử lý số liệu trên On-chip 80C51

Số liệu từ bộ đếm xung phương vị được đưa vào hệ vi xử lý On-chip qua kênh 12 bit, 8 bit thấp đưa qua cổng P1 còn 4 bit cao đưa qua cổng P3. Số liệu phương vị trung bình của On-chip được đưa vào bộ đệm cho máy tính qua kênh 12 bit, 8 bit thấp đưa qua cổng P0 còn 4 bit cao đưa qua cổng P2.

Tín hiệu điều khiển khác được lấy ra từ các chân tín hiệu điều khiển còn lại của hệ vi xử lý On-chip như thể hiện trên hình 11.16. Để tạo nhịp cho hệ vi xử lý On-chip, phải sử dụng dao động thạch anh mắc vào chân X1X2 có trị số 12 MHz. Hệ sử dụng bộ nhớ ROM và RAM nội trú nên chân /EA được nối lên +Vcc.

Bộ đếm xung phương vị 12 bit được tổ chức trên cơ sở 3 chip đếm 74193 theo phương pháp mắc nối tiếp. 8 bit có trọng số thấp được đưa vào hệ vi xử lý On-chip qua cổng P1 còn 4 bit có trọng số cao của bộ đếm sẽ đưa qua cổng P3 để vào hệ vi xử lý On-chip. Bộ đếm sẽ nhận tín hiệu xung từ đĩa tạo xung phương vị khi anten quay. Cứ 4096 xung thì hết một vòng. Tại thời điểm này đĩa tạo xung phương vị sẽ tạo thêm một xung đánh dấu đặc biệt là xung phương vị bắc để báo kết thúc một vòng kiểm soát (360 độ) và khởi đầu một chu kỳ mới. Xung đặc biệt này được dùng để RESET bộ đếm 12 bit này.

Bộ đệm 12 bit chứa giá trị phương vị trung bình cho máy tính PC được tổ chức trên cơ sở hai chip 74373. 8 bit thấp được lấy ra từ cổng P0 của On-chip, còn 4 bit cao được lấy ra từ cổng P2 của On-chip. Điều khiển chốt từ chân tín hiệu P2.6 qua chip 125 đưa mức 1 logic vào chân LE (load enable) của 373. Máy tính PC sẽ đưa tín hiệu đọc vào qua chân /OE (output enable) của 373 khi nhận được ngắt.

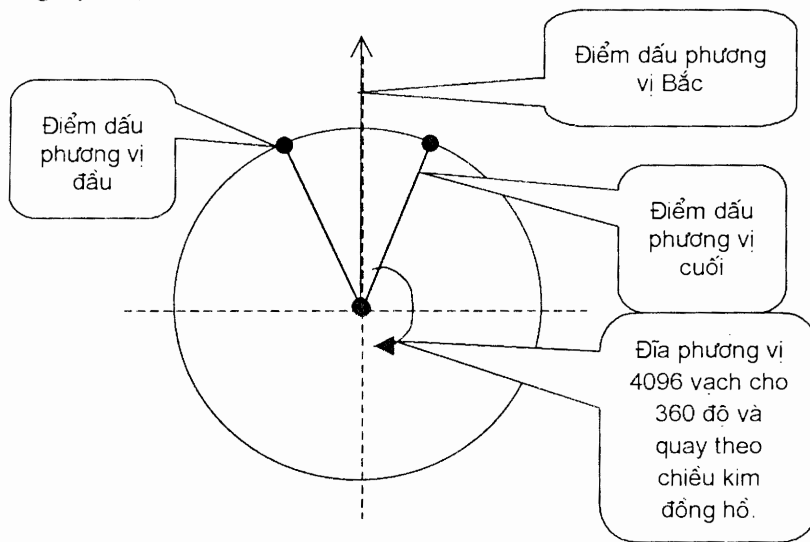
Xây dựng phần mềm của hệ quan sát mục tiêu

Trong thuật toán của phần mềm điều khiển sẽ thực hiện việc kiểm soát hai quá trình nối tiếp nhau. Quá trình thứ nhất là quá trình phát hiện điểm dấu đầu của mục tiêu khi điều kiện có từ 3 xung phản xạ có mức logic 1 trở lên trong 5 lần trích mẫu được thoả mãn. Quá trình thứ hai là quá trình phát hiện điểm dấu kết thúc quan sát mục tiêu khi điều kiện có từ 3 xung phản xạ có mức logic 0 trở lên trong 5 lần trích mẫu được thoả mãn. Mặt khác các thời điểm xuất hiện điều kiện trích mẫu được thoả mãn theo thuật toán trên là không biết trước nên phải dùng cơ chế ngắt của hệ vi xử lý On-chip để thực hiện việc kích hoạt các chương trình con xử lý các quá trình tương ứng ở trên. Quá trình thứ nhất chỉ cần lấy giá trị phương vị khởi đầu lưu trữ vào bộ nhớ trong nên không cần nhiều thời gian. Vì vậy nó được nối với đầu ngắt IT1 là đầu ngắt có mức ưu tiên thấp (ở chế độ mặc định). Quá trình thứ hai không những cần lấy giá trị phương vị kết thúc quan sát để lưu trữ vào bộ nhớ trong mà còn phải thực hiện phép tính lấy trung bình cộng của phương vị mục tiêu và cất giữ vào bộ đệm trung gian cho máy tính PC nên cần nhiều thời gian hơn so với tiến trình đầu. Vì vậy nó được nối với đầu ngắt IT0 là đầu ngắt có mức ưu tiên cao nhất (ở chế độ mặc định).

Lưu ý rằng khi thực hiện phép tính lấy trung bình cộng của phương vị mục tiêu phải xử lý tình huống khi giá trị phương vị cuối lại nhỏ hơn giá trị phương vị đầu (hình 11.17). Khi đó, giá trị phương vị trung bình được tính bằng cách lấy bù:

PV tr.b = (PV cuối + 4096 + PV đầu)/2.

Nếu kết quả PV tr.b < 4096 thì giữ nguyên giá trị kết quả (điểm tính toán nằm bên trái điểm dấu phương vị bắc). Nếu kết quả PV tr.b>= 4096 thì giá trị kết quả phải trừ bớt đi 4096 (điểm tính toán nằm chính giữa hoặc nằm bên phải điểm dấu phương vị bắc).

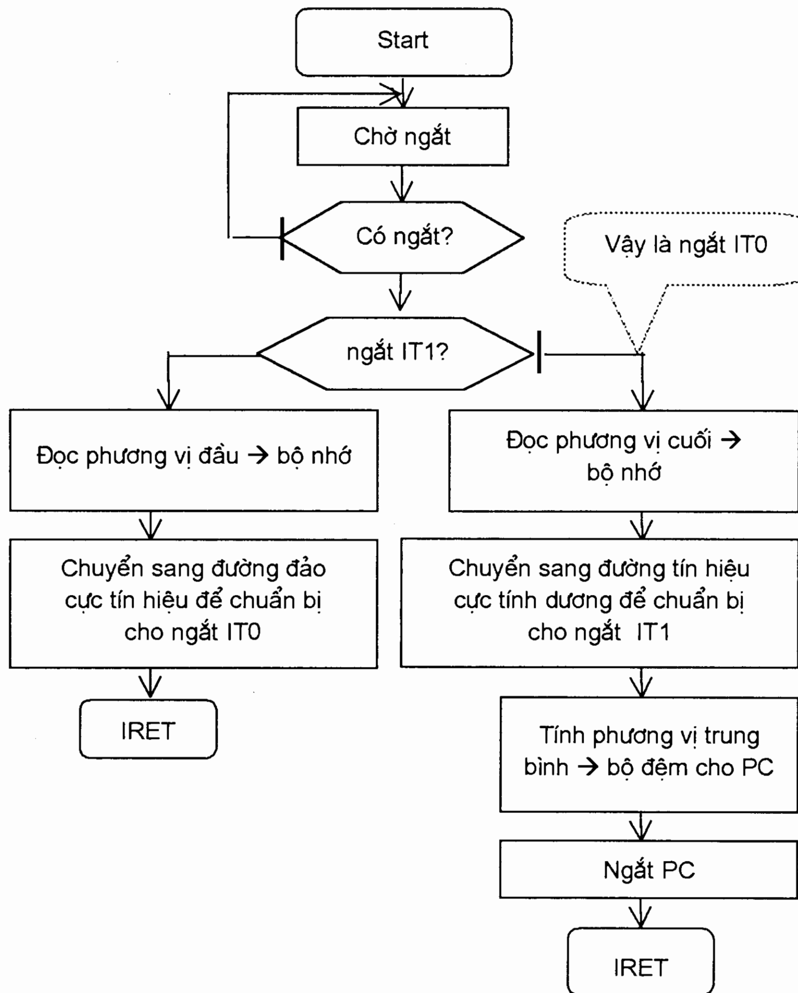


Hình 11.17. **Xử lý trường hợp đặc biệt khi tính PV trung bình**

Căn cứ vào các phân tích trên, lưu đồ thuật toán tối ưu cho phần mềm điều khiển được thể hiện trên hình 11.18. Trong tiến trình tính phương vị trung bình để đưa vào bộ đệm cho PC cần xử lý cả tình huống đặc biệt khi xảy ra như hình 11.17 mô tả.

Với thuật toán và lưu đồ thuật toán đã xây dựng trên, chương trình nguồn ASSEMBLY cho hệ vi xử lý này được viết như sau: đầu tiên là gán giá trị địa chỉ ngăn nhớ cho các vector ngắt theo quy định rồi chuyển điều khiển tới phần khởi động các thành phần trong hệ vào trạng thái ban đầu cho đúng với chức năng của chúng như tổ chức ngăn xếp bắt đầu từ địa chỉ 6FH; lập bit P2.5-định hướng thu; cho phép ngắt ngoài được hoạt động; lập mức ưu tiên cho ngắt.

Phần chương trình chính chỉ có nhiệm vụ chờ tín hiệu ngắt từ bên ngoài nên nó không làm gì cả và được thể hiện bằng vòng lặp WAIT.



Hình 11.18. **Lưu đồ thuật toán cho phần mềm điều khiển**

Chương trình con phục vụ ngắt được xây dựng đúng chức năng yêu cầu. Chương trình con NGAT0 được bắt đầu từ địa chỉ 0003h bằng lời gọi ACALL tới chương trình, mà nó được tổ chức như chương trình bình thường. Tương tự như vậy, chương trình con NGAT1 được bắt đầu từ địa chỉ 0013h bằng lời gọi ACALL tới chương trình, mà nó cũng được tổ chức như chương trình bình thường.

```
;------------------------------------------------------------
; Phần khai báo các hằng ký hiệu
;------------------------------------------------------------
VECTOR0 EQU 0003H
VECTOR1 EQU 0013H
PV_Start0  EQU R0              ; byte thấp phương vị đầu
PV_Start1  EQU R1              ; byte cao phương vị đầu
```

```
PV_End0  EQU R2              ; byte thấp phương vị cuối
PV_End1  EQU R3              ; byte cao phương vị  cuối
Tuyen_Thuan  EQU    P2.5      ; cho ngắt INT1
Tuyen_Nghich  EQU    P2.4     ; cho ngắt INT0
WriteToBUF   EQU P2.6   ; lệnh ghi vào đệm cho PC
IntPC    EQU    P3.0         ; yêu cầu ngắt PC
;-----------------------------------------------------------------
;Bắt đầu chương trình
;-----------------------------------------------------------------
BEGIN: AJMP START
;----------------------------------------------------------------
;Ngắt ở địa chỉ 0003h - Ngắt IT0
;----------------------------------------------------------------
ORG VECTOR0
lJMP NGAT0 ; gọi chương trình con phục vụ ngắt 0
;----------------------------------------------------------------
; ngắt  địa chỉ 0013h - ngắt IT1
;----------------------------------------------------------------
ORG VECTOR1
lJMP NGAT1   ; gọi chương trình con phục vụ ngắt 1
;------------------------------------------------------------------
START:
          MOV SP, #6FH ; đỉnh ngăn xếp được gán = 6Fh
          MOV P2, #00H ; cho cổng P2 = 00h
          SETB Tuyen_Thuan  ; lập bit P2.5-định hướng thu
          SETB EX0  ; cho phép ngắt ngoài IT0 được hoạt động
          SETB EX1  ; cho phép ngắt ngoài IT1 được hoạt động
          CLR C           ; xoá bit cờ
          SETB EA   ; lập bit EA = 1-cho phép cơ chế ngắt hoạt động
WAIT:
          NOP ;đợi ngắt
          AJMP WAIT
;-----------------------------------------------------
```

```
; Bắt đầu phần xử lý của ngắt INT0
;-----------------------------------------------------
NGAT0:
        CLR EA ; cấm ngắt EA = 0
        MOV PV_End0, P1 ; R2<---- 8 bit phương vị P1
        MOV A, P3   ; Acc<--- 4 bit phương vị P3
        RR A        ;dịch phải Acc 4 bit bằng 4 lệnh RR
        RR A
        RR A
        RR A
        ANL A, #0FH ;lọc bỏ 4 bit cao
        MOV PV_End1, A    ; chuyển vào R3-chứa 4 bit cao
;-----------------------------------------------------
;bắt đầu trừ R3 với R1
;-----------------------------------------------------
        CLR C
        SUBB A, PV_Start1  ; a<---R3-R1
        JC XLPVBAC ; nhảy tới nhãn xử lý phương vị Bắc
;-----------------------------------------------------
; lấy phương vị trung bình
;-----------------------------------------------------
        MOV A, PV_Start0  ; lấy byte thấp của PV đầu
        CLR C   ; xoá  CF → 0
        ADD A, PV_End0  ; cộng với byte thấp của PV cuối
        MOV PV_Start0, A ; tổng 2 byte thấp → R0
        MOV A,  PV_Start1; lấy byte cao của PV đầu
        ADDC A, PV_End1  ; cộng với byte cao của PV cuối
        CLR C  ; xoá  CF → 0
        RRC A         ; dịch phải để chia 2
        MOV PV_Start1, A
        MOV A, PV_Start0
        RRC A; cờ CF của byte cao nếu có thì được → bit Acc7
        MOV PV_Start0 , A; lúc này R1R0 chứa giá trị phương vị trung bình
        AJMP OUTPORT
```

```
XLPVBAC:
          ADC PV_End0 , #00h
          ADC PV_End1, #10h; cộng bù 4096
          ADC PV_End0, PV_Start0  ; cộng 2 byte thấp
          ADC PV_End1, PV_Start1  ; cộng 2 byte cao→ R3R2
                                  ;chứa tổng với số bù
          RRC PV_End1; dịch phải để chia 2
          RRC PV_End0
          MOV PV_Start1, PV_End1 ;gửi R3→R1
          SUBB PV_End1, #10h; trừ đi 4096
          JC Left  ; có nhớ, tức R3<4096 nên nằm ở trái PV bắc
          MOV PV_Start1, PV_End1 ; không nhớ, tức ở bên phải PV bắc:
                                  ;R3-4096 chính là kq
LEFT: MOV A, PV_End0; lưu ý là R1R2 là kq trong trường hợp
                                  ;này: khi R3<4096
OUTPORT:
          MOV P0, A; đưa byte phương vị thấp ra byte thấp của đệm PC
          MOV A, PV_Start1
          CLR  Huong_Thuan ; ~anl a,#11101111b ; duy trì
                              ;hướng thu đúng đầu chùm tiếp theo
          SETB Huong_nghich ; cấm hướng nghịch
          MOV P2, A           ; đưa ra 4 bit phương vị cao cho đệm PC
          CLR  WriteToBUF   ; P2.6 mở bộ đệm PV sang phía PC
          SETB EA             ;cho phép ngắt tiếp
          SETB IntPC          ; P3.0 đưa yêu cầu ngắt vào PC
          SETB  INT0          ; đưa đầu ngắt IT0- P3.2 lên cao tạo điều kiện
                              ;cho ngắt IT0 ở lần sau
          RETI
;----------------------------------------------------
; Bắt đầu phần xử lý của ngắt INT1
;----------------------------------------------------
NGAT1:
          CLR P3.0 ;huỷ yêu cầu ngắt vào PC
```

```
        CLR EA
        CLRB Huong_ nghich ; chuyển hướng thu cuối chùm
        SETB Huong_ Thuan ; cấm hướng thuận- thu đầu chùm
        MOV PV_Start0, P1    ; thu byte PV thấp vào R0
        MOV A, P3
        RR A ;dịch phải 4 bit
        RR A
        RR A
        RR A
        ANL A, #0FH; lọc bỏ 4 bit cao
        MOV PV_Start1, A   ;lưu 4 bit cao vào R1. Bây giờ R1R0
                           ;chứa giá trị  PV đầu chùm 12 bit
        SETB INT1; đưa đầu ngắt INT1- P3.3 lên cao
        NOP
        NOP
        SETB EA ; lại cho phép ngắt hệ on chip
        RETI
END.
```

Chương 12
HỆ XỬ LÝ SONG SONG

Trong các chương trước ta đã thấy rõ những ưu điểm to lớn của kỹ thuật vi xử lý trong các lĩnh vực kỹ thuật hiện đại. Song cũng phải nhận thấy một nhược điểm là trong xử lý các quá trình biến đổi nhanh hoặc các thao tác tính toán phức tạp thì tốc độ của hệ vi xử lý xây dựng theo nguyên tắc trên không đáp ứng được. Giải pháp khả thi hơn cả là xây dựng hệ xử lý song song dựa trên sự tổ hợp các bộ vi xử lý CPU cùng hướng tới thực hiện một nhiệm vụ chung.

Hệ xử lý song song là cấu trúc Automat hữu hạn cho phép thực hiện phép xử lý đồng thời nhiều tiến trình mà hệ quả là tốc độ xử lý chung của hệ thống có thể cao hơn tốc độ xử lý của các thành phần cấu thành hệ thống.

12.1. PHÂN LOẠI KIẾN TRÚC XỬ LÝ SONG SONG

Bản thân hệ xử lý song song là cấu trúc Automat hữu hạn cho phép thực hiện phép xử lý nhiều quá trình xếp chồng về mặt thời gian. Có ba kiểu kiến trúc chính là: kiến trúc kiểu PIPELINE, kiến trúc kiểu ma trận và kiến trúc kiểu đa CPU.

Kiểu Pipeline thực hiện việc xếp chồng các thao tác tính toán trung gian về mặt thời gian và tiến hành xử lý đồng thời chúng trong chu kỳ lệnh.

Kiểu ma trận sử dụng tập hợp các khối tính toán theo phương thức đồng bộ cho phép thực hiện việc xử lý song song theo không gian nhiều chiều.

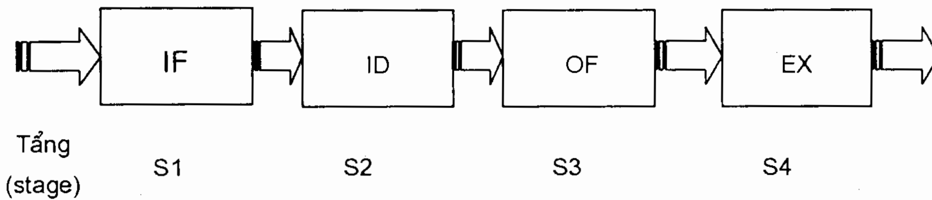
Kiểu đa CPU thực hiện phép xử lý các quá trình song song không đồng bộ thông qua tập hợp các CPU có mối quan hệ ràng buộc sao cho chúng có thể chia xẻ tài nguyên của hệ thống mà không gây ra tranh chấp, xung đột.

12.2. KIẾN TRÚC KIỂU PIPELINE

12.2.1. Cấu trúc của hệ xử lý PIPELINE

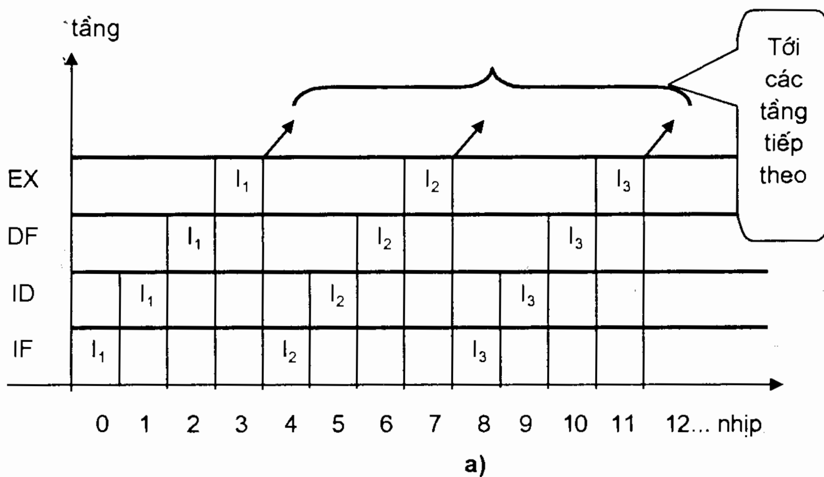
Trong hệ xử lý song song theo kiến trúc PIPELINE, các thao tác trong một chu kỳ lệnh cần chia ra thành các chức năng con theo mối quan hệ duy nhất là trình tự thời gian. Mỗi khoảng thời gian trong trình tự là một tầng chức năng Stage. Hình 12.1 mô tả kiến trúc PIPELINE 4 tầng. Mỗi tầng có chức năng riêng:

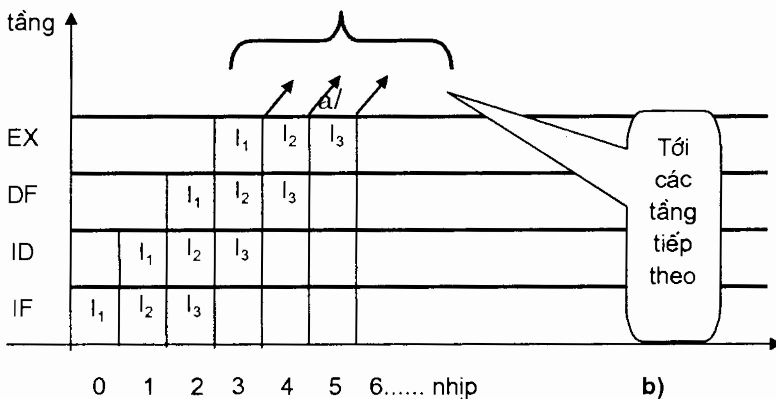
- Nhận lệnh IF (Instruction fetch)
- Giải mã lệnh ID (Instruction decoding)
- Nhận toán hạng OF (Operand fetch)
- Thực hiện lệnh EX (Execution)



Hình 12.1. **Kiến trúc PIPELINE 4 tầng**

Về nguyên tắc, xử lý theo kiểu PIPELINE sẽ tăng tốc độ xử lý lên K lần (với K là số tầng của Pipeline) như được mô tả trên hình 12.2. Song hệ xử lý là hệ kỹ thuật nên tốc độ các thành phần khác nhau như bộ nhớ, cổng giao tiếp, thao tác rẽ nhánh làm tốc độ không còn như lý thuyết.





Hình 12.2. **Hiệu ứng PIPLINE**

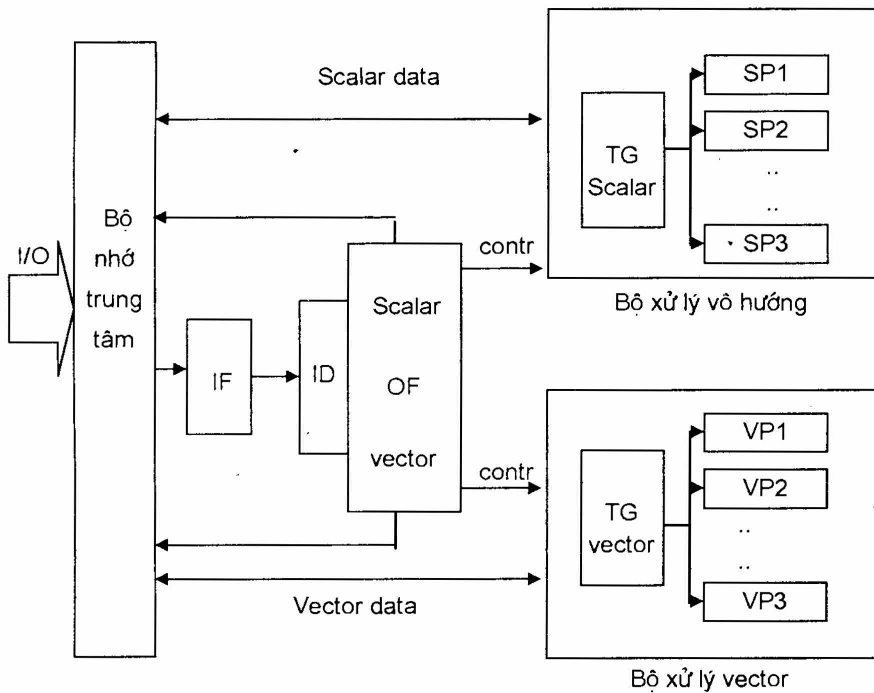
a) xử lý không xếp chồng thời gian; b) xử lý có xếp chồng thời gian.

So sánh hai trường hợp của kiến trúc PIPLINE trên hình 12.2 ta thấy rõ trường hợp xếp chồng về thời gian có thời gian thực hiện ít hơn so với trường hợp không xếp chồng về thời gian.

Với kiến trúc của kiểu PIPLINE ta nhận thấy dễ dàng Vector hoá quá trình xử lý thông tin và do vậy việc xử lý song song sẽ được kiểm soát một cách chặt chẽ. Những cấu trúc điển hình cho kiến trúc này là máy tính Cray-1, VP-200 cho ta những khái niệm cơ bản về loại hình này.

Trong cấu trúc của các máy tính này, khối chuẩn bị lệnh bao gồm ba tầng mà mỗi tầng đều có kiểu Pipeline. Riêng tầng OF gồm hai phần độc lập nhau: một cho toán hạng vô hướng, một cho toán hạng Vector. Các thanh ghi dùng trong khối Scalar có cấu trúc đơn giản giống như các thanh ghi thông thường, còn các thanh ghi Vector phức tạp hơn vì nó phải chứa nhiều thành phần. Ví dụ thanh ghi vector của máy tính Cray-1 có 64 thành phần, mỗi thành phần 64 bit, vị chi là 4096 bit.

Dữ liệu dạng vô hướng hay dạng vector đều có thể xuất hiện ở dạng dấu phẩy tĩnh hay dấu phẩy động. Các cấu trúc pipeline vô hướng ở tổ chức phần cứng và có nguyên lý điều khiển riêng.



Hình 12.3. **Cấu trúc của khối chuẩn bị lệnh trong**

Các hệ xử lý vector hiện đại được tăng cường đáng kể bởi các bộ xử lý mạnh để điều phối sự phối hợp (mixture) các lệnh dạng vector và các lệnh vô hướng.

12.2.2. Nguyên tắc của phương pháp xử lý vector trong PIPLINE

Pipeline là kiến trúc cho phép xử lý theo nguyên tắc xếp chồng các chức năng về thời gian nên để tạo hiệu ứng pipeline có hiệu quả, phải phân chia nhiệm vụ đầu vào thành trình tự các nhiệm vụ con. Khái quát hơn đó là việc sử dụng tối ưu phương pháp phân rã hàm chức năng trong lý thuyết Điều khiển học để tách chức năng lớn thành tập hợp các chức năng con trong mối quan hệ định lượng có thể kiểm soát được. Trường hợp đơn giản nhất của mối quan hệ ràng buộc là quan hệ tuyến tính theo thời gian, tức là tạo ra chức năng con theo kiểu xếp hàng theo thời gian. Như vậy:

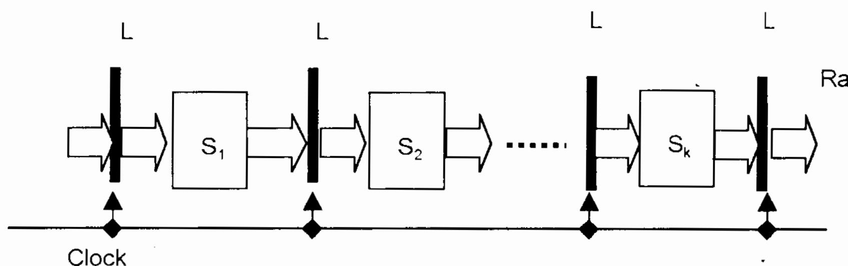
- Mỗi chức năng con được thực hiện trên một cấu trúc riêng - là một tầng trong dòng xử lý của pipeline.
- Mỗi chức năng con sẽ được cơ chế điều khiển phân luồng vào pipeline dựa vào các thuộc tính của chính nó.
- Khởi động toàn hệ thống để xử lý theo nguyên lý xếp chồng.

Cấu trúc pipeline tuyến tính

Nhiệm vụ T (Task), theo phương thức trên được chia thành tập hợp con

{T1, T2,...,Ti, Tj, Tk} với k là số lượng tử thời gian mà nhiệm vụ T thực hiện trọn vẹn chức năng của mình. Nói một cách tổng quát: mỗi chức năng con Tj bất kỳ chỉ có thể bắt đầu khi mà một chức năng Ti (i < j) nào đó đã kết thúc.

Cấu trúc của pipeline tuyến tính được thể hiện trên hình 12.4.



Hình 12.4. **PIPELINE tuyến tính: L -Latch (bộ chốt), Si -Stage (tầng)**

Khi cấu trúc trên hoạt động, ta có các đại lượng thời gian xác định như sau:

T_L- Thời gian giữ chậm khi thao tác chốt dữ liệu vào các chốt L.

T_i - Thời gian cần cho việc thực hiện thao tác của tầng chức năng S_i trong hệ.

Như vậy, một chu kỳ nhịp được tính theo biểu thức sau:

$$\tau = \max\{\tau_i\}\Big|_1^k + \tau_L = \tau_{max} + \tau_L$$

Tần số làm việc được tính $f = 1/\tau$.

Thí dụ, cho:

$\tau_{max} = 4$ ns $= 4.10^{-9}$ s.

$\tau_i = 1$ ns $= 1.\ 10^{-9}$ s.

sẽ tính được tần số làm việc:

$f = 1/5.10^{-9} = 2.10^{-1}.10^{9} = 2.10^{8} = 200$Mhz.

Vậy là nếu hệ có cấu trúc kiểu pipeline có K tầng thì để xử lý n nhiệm vụ nó cần số lượng nhịp là:

$$T_{Kp} = K + (n - 1).$$

Như vậy, nhiệm vụ đầu thực hiện hết K nhịp xung Clock, còn (n – 1) nhiệm vụ còn lại thì mỗi nhiệm vụ chỉ thực hiện trong một chu kỳ nhịp nhờ hiệu ứng xếp chồng thời gian.

Trong khi đó hệ xử lý thông thường (không có hiệu ứng pipeline) cần số lượng nhịp là:

$T_{k0} = n.\ K$ nhịp xung Clock.

Sự chênh lệch về tốc độ có thể xác định được bằng tỉ số:

$$\eta_k = \frac{T_{k0}}{T_{Kp}} = \frac{(n.K)}{(K + (n - 1))}$$

Nếu K càng lớn thì sự chênh lệch càng cao và có thể coi hệ pipeline có tốc độ lớn gấp K lần với hệ bình thường.

Thí dụ minh hoạ cho hệ pipeline tuyến tính

Cho 2 số dấu phảy động $A = a \times 2^p$

$B = b \times 2^q$

$C = A+B = c \times 2^r = d \times z^s$

Với $r = \max(p,q)$

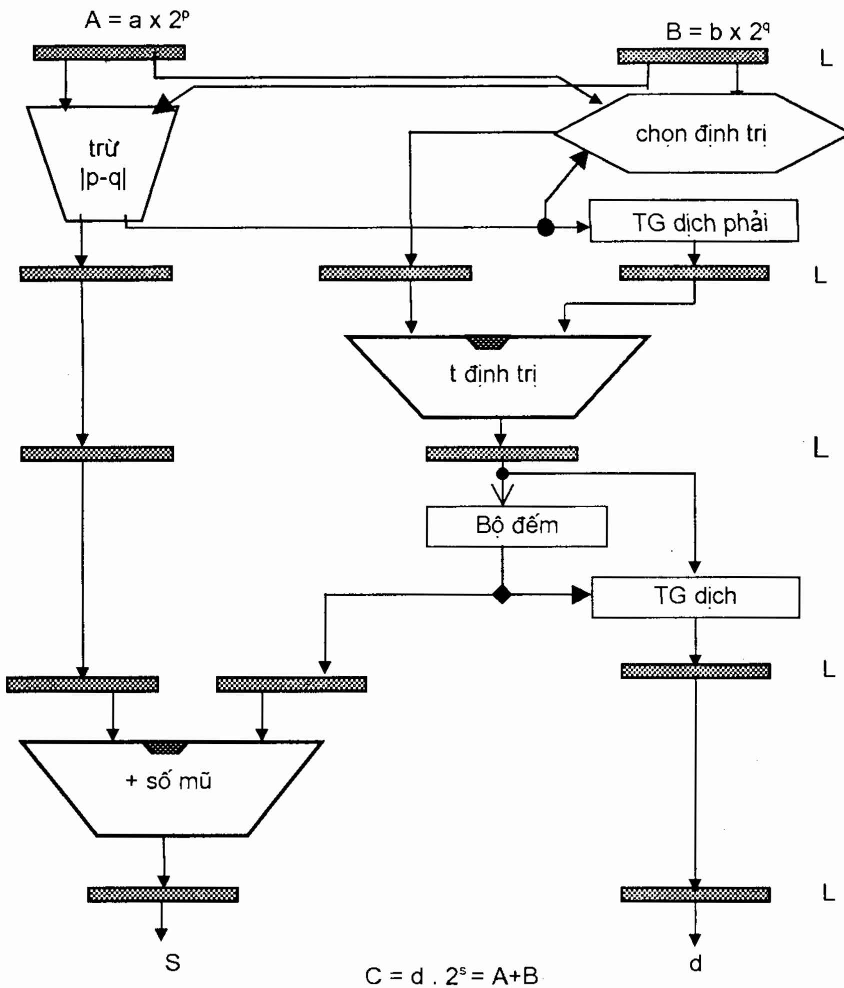
$0,5 \le d < 1$

Thuật toán: 1. So sánh p, g theo $\max(p, g) = t = |p - g|$ (hình 12.5)

Dịch phải số có số mũ nhỏ đi t bước.

Cộng phần định trị của hai số.

Chuyển dạng hợp thức của kết quả.



Hình 12.5. **Tính cộng 2 số dấu phảy động trên hệ pipeline tuyến tính**

12.2.3. Kiến trúc pipeline có khả năng rẽ nhánh

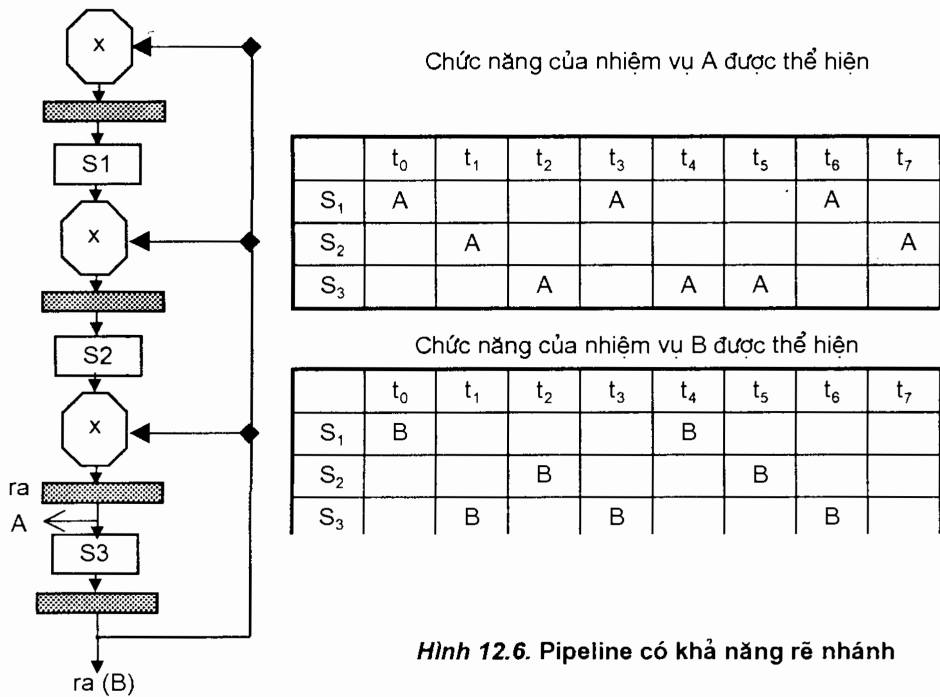
Một pipeline hoạt động mềm dẻo phải có khả năng cho phép các nhiệm vụ con bỏ qua một số tầng phía trước hoặc quay lại các tầng phía sau, tức là có khả năng nhảy xuôi và nhảy ngược. Khả năng này tuân thủ theo điều kiện:

Nhảy xuôi Si $\rightarrow$ Sj sao cho $j \geq i+2$.

Nhảy ngược Si $\rightarrow$ Sj sao cho $j \leq i$

Hoạt động của pipeline như vậy đòi hỏi một bảng mô tả các chức năng con và lưu đồ chứa các tiến trình như được minh hoạ trên hình 12.6.

Khi xử lý theo kiểu vector yêu cầu bộ nhớ của hệ phải có tổ chức sao cho không xảy ra xung đột, tranh chấp giữ các tiến trình. Tham số quan trọng cần tính đến là độ rộng của kênh thông tin (tính theo byte hay words). Cùng với tham số trên thì số lượng các Modul nhớ độc lập - phương pháp địa chỉ hoá và điều kiện hoá chúng cũng có vai trò quan trọng.



Chức năng của nhiệm vụ A được thể hiện

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	A			A			A	
S_2		A						A
S_3			A		A	A		

Chức năng của nhiệm vụ B được thể hiện

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
S_1	B				B			
S_2			B			B		
S_3		B		B			B	

Hình 12.6. **Pipeline có khả năng rẽ nhánh**

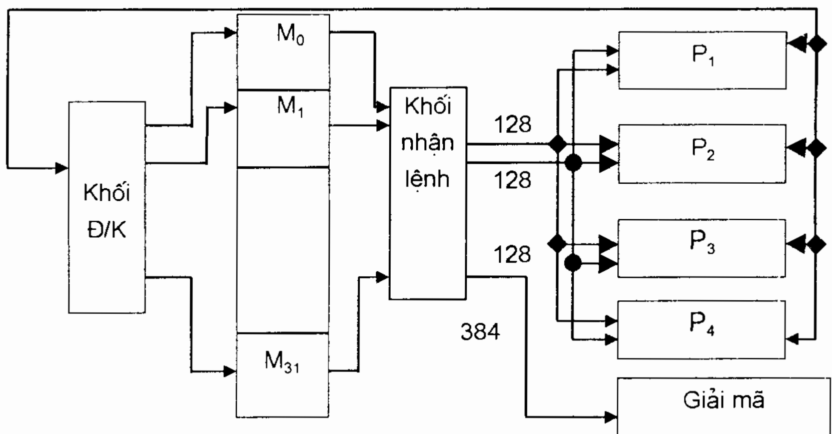
Hình 12.7 mô tả phương pháp tổ chức kênh thông tin cho hệ có 4 Pipeline. Cách tính số lượng bit cần thiết cho kênh:

2 operand × 32 bit (vào các P)
1 operand × 32 bit (cất vào bộ nhớ)
1 kênh × 32 bit

(4 × 32 bit) × 4 pipline → 4 × 128 bit Bus

Nếu thời gian quy chiếu bộ nhớ 1.28 μs và nếu chu kỳ làm việc của pipline là 40 μs thì cần số lượng các modul nhớ là:

SL modul M = 1.28μs/40μs = 32M $\{M_0, M_1, \ldots\ldots, M_{31}\}$



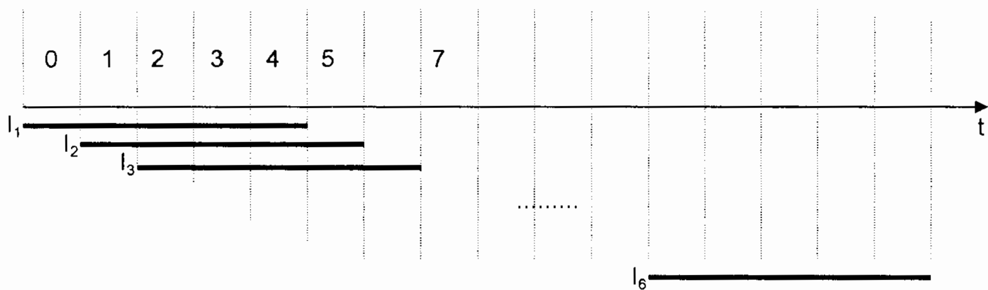
Hình 12.7. **Tổ chức kênh thông tin trong hệ PIPELINE**

12.2.4. Tổ chức hệ xử lý pipeline

Khi tổ chức hệ xử lý cần thống kê các thao tác tính toán và chức năng theo tỉ lệ tương đối vào một bảng như minh hoạ trên bảng 12.1 và hình 12.8.

Bảng 12.1

Tính toán/ nạp lệnh, nạp toán hạng	Cất giữ	Rẽ nhánh có điều kiện		
60%	20%		12% thoả mãn	8%



Hình 12.8. **Hiệu ứng pipline bị huỷ bỏ khi gặp lệnh nhảy**

Gọi p- Khả năng xuất hiện các lệnh rẽ nhánh (theo bảng 12.1 là 20 phần trăm).

q- Khả năng xuất hiện thoả mãn điều kiện (theo bảng 12.1 là 12 phần trăm).

n- Số lượng lệnh chờ thực hiện ở pipeline.

Vậy tích n.p.q là số lượng lệnh thoả mãn điều kiện rẽ nhánh.

Thời gian bổ xung cho việc thực hiện 1 lệnh rẽ nhánh là (K-1)/K với K là số tầng.

Số lượng chu kỳ lệnh cần có để n lệnh được thực hiện là

$$\frac{(K+(n-1))}{K}+\frac{(npq)(K-1)}{K}$$

Tốc độ thực hiện được tính khi n rất lớn

$$\text{Lim}_{n\to\infty}=\frac{n}{\left(\frac{(K+n-1)}{K}+\frac{np.q(K-1)}{K}\right)}=\frac{K}{(1+p.q.(K-1)}$$

12.3. LẬP TRÌNH CHO HỆ XỬ LÝ SONG SONG

Khi xây dựng chương trình cần sử dụng công cụ là ngôn ngữ tương ứng để chuyển ý đồ thành các lệnh cho hệ xử lý hiểu- lệnh mã máy.

Giả sử cần thực hiện phép cộng 2 đại lượng A(I) và C(I) đặt trong A(I)

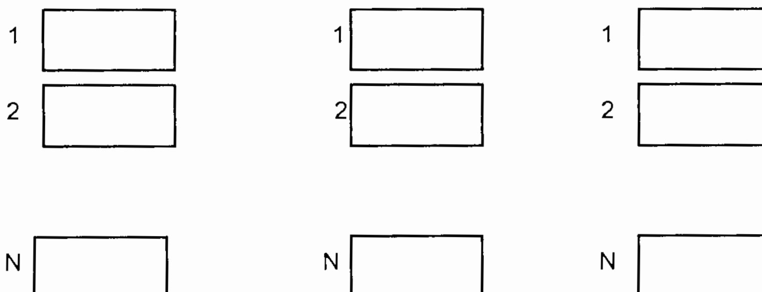
$\{I = \overline{1, N}\}$

A(I)=B(I)+C(I)

B(I)=2*A(I+1)   Với $I = \overline{1, N}$

Thao tác với hệ vô hướng:

```
        NTIALIZE I=1
10      READ B(I)
        READ C(I)
        ADD B(I)+C(I)
        STORE A(I) ← B(I)+C(I)
        READ A(I+1)
        MULTIFY  2 ↓A(I+1)
        STORE   B(I) ← 2↓ A(I+1)
        INCREMENT I ←I+1
        IF  I ≤ N  GO TO 10
        STOP
```



Đối với hệ vector :

A(1:N) = B(1:N)+C(1:N)

TEMP (1:N) = A(2:N+1)

B(1,N) = 2↓TEMP(1:N)

A(1:N) trỏ tới mảng vector N phần tử {A(1),A(2),...,A(N)} TEMP (1:N) là lệnh cho phép thực hiện việc triển khai quá trình vector hóa.

Như vậy chương trình trong hệ vô hướng sẽ phải thực hiện nhiều lần lặp lại, còn trong hệ xử lý vector điều này bị loại bỏ bởi tổ chức phần cứng.

Để xây dựng cơ chế điều khiển hệ xử lý song song cần có các thành phần sau:

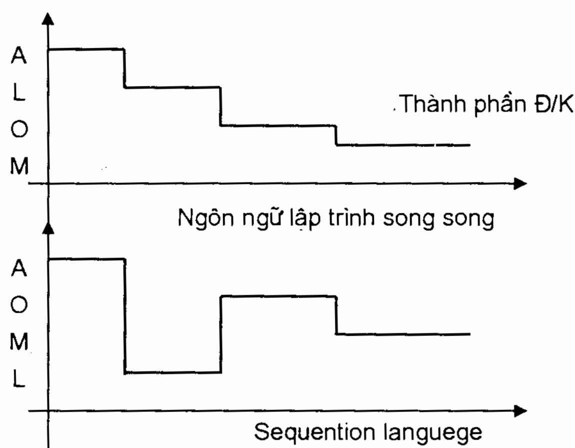
A -Thuật toán xử lý song song (Algorinthm).

L - Ngôn ngữ (Language).

O - Mã đối tượng (Object Code).

M -Mã máy (Machine Code).

Hình 12.9 là sự so sánh phương thức lập trình song song và lập trình thông thường.



Hình 12.9. **So sánh phương thức lập trình song song và lập trình thông thường**

12.4. HỆ XỬ LÝ SONG SONG KIỂU ĐA CPU

Hệ đa CPU có thể đặc trưng bởi hai thuộc tính:

- Là hệ xử lý độc lập bao gồm nhiều CPU;
- Các CPU có thể liên hệ và hợp tác hoạt động theo mức độ khác nhau trong thực hiện nhiệm vụ được giao.
- Sự liên lạc giữa các CPU thực hiện theo phương thức:

Gửi thông báo cho nhau.

Chia xẻ bộ nhớ chung.

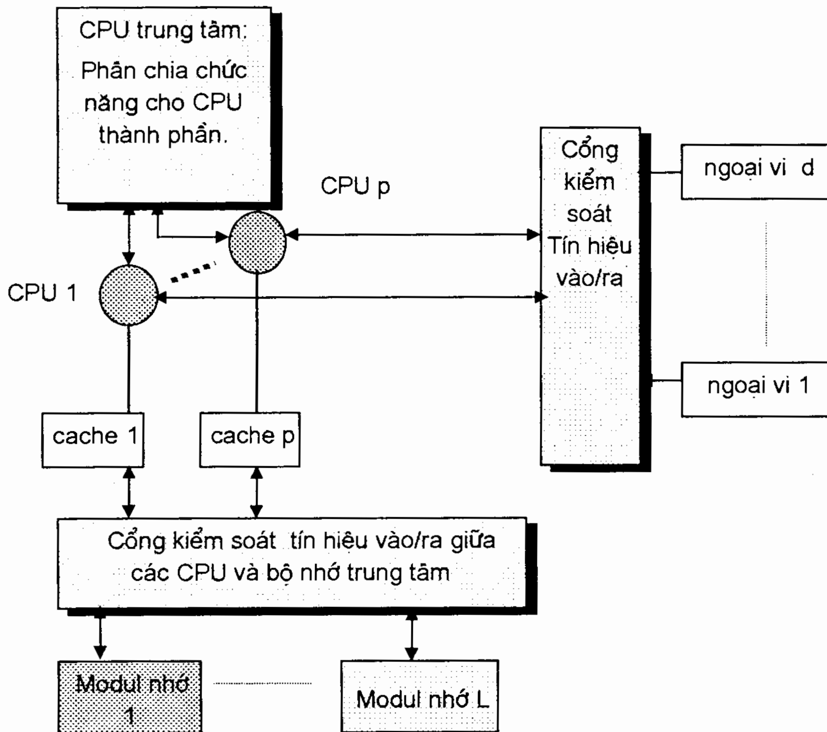
Như vậy sẽ có các thao tác đồng thời xảy ra trong hệ vì vậy cần có khối điều khiển thao tác hệ thống cung cấp khả năng phối hợp các CPU và chương trình của chúng ở các quá trình xử lý, thiết lập dữ liệu, mức và quyền hạn xử lý chúng (Hệ Denel Hep).

Khối điều khiển phải có khả năng phân tích từng cụm lệnh để hình thành các tiến trình con có thuộc tính song song rồi phân chia cho các CPU thành phần một cách tối ưu và cung cấp phương thức phối hợp các CPU này ở mọi giai đoạn hoạt động như giai đoạn xắp xếp, thiết lập dữ liệu, kiểm soát mức và quyền hạn truy nhập chúng.

12.4.1. Cấu trúc của hệ xử lý song song đa CPU

Với cách đặt vấn đề như vậy, chúng ta sẽ xây dựng hệ xử lý song song gồm p CPU tạo thành không gian vector tuyến tính p chiều. Trong quá trình thao tác, mỗi

CPU có thể phải truy cập nhiều lần tới bộ nhớ trung tâm để cập nhật số liệu và do đó làm tăng thời gian trễ của hệ thống. Để giảm thời gian trễ này ta tổ chức cơ cấu Cache cho mỗi CPU. Như vậy, chỉ ở giai đoạn cuối các CPU mới phải truy cập tới bộ nhớ trung tâm, điều này làm giảm đáng kể thời gian xắp hàng ở cổng vào/ra giữa các CPU với bộ nhớ trung tâm. Sơ đồ chức năng của hệ xử lý này có dạng như hình 12.10.



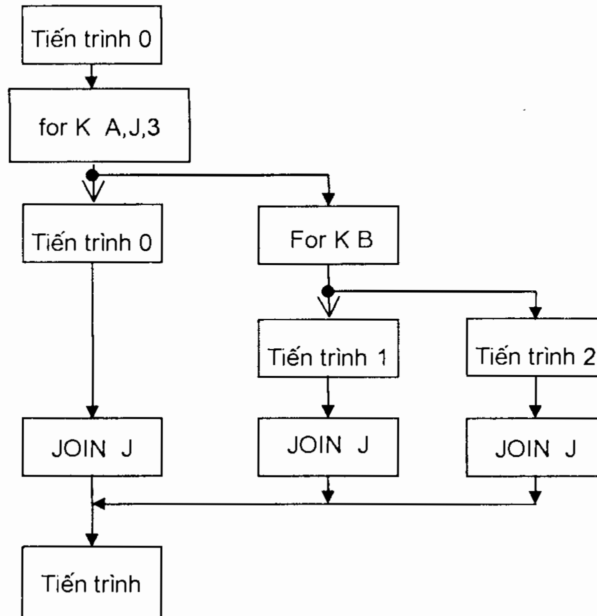
***Hình 12.10.* Tổ chức phần cứng của hệ xử lý song song p CPU**

12.4.2. Lập trình cho hệ xử lý song song đa CPU

Tiêu chuẩn cơ bản để CPU trung tâm có thể phân tích và phân chia chức năng chính thành các chức năng con là tiêu chuẩn Beinstein. Điều này có nghĩa là nếu thoả mãn các tiêu chuẩn song song thì hệ thống có thể khởi đầu một tiến trình mới mà các tiến trình cũ vẫn đang tiếp diễn. Lúc này CPU trung tâm sẽ sử dụng phát biểu FORK và JOIN để mô tả hoạt động.

FORK dùng để phát sinh một tiến trình mới: FORK A sẽ khởi đầu một tiến trình khác bắt đầu từ địa chỉ A và thực hiện tiếp tiến trình hiện hành; FORK A, J cũng thực hiện giống như FORK A kèm theo việc tăng giá trị của bộ đến tại địa chỉ J; FORK A, J, N cũng thực hiện giống như FORK A kèm theo việc đặt bộ đến tại J lên giá trị N.

Đối với mọi dạng thức của FORK thì phát biểu JOIN chỉ có một dạng thức là JOIN J. Khi thực hiện phát biểu này thì JOIN J sẽ giảm nội dung bộ đếm J đI một đơn vị. Nếu kết quả bằng 0 thì một tiến trình tạI J+1 sẽ được khởi đầu. Nói cách khác các bộ xử lý thực hiện các tiến trình trước sẽ được hoặc là giải phóng hoàn toàn hoặc là chuẩn bị để thực hiên các tiến trình mới (hình 12.11).



Hình 12.11. Cấu trúc FORK và JOIN

Ngôn ngữ lập trình song song ***Dikstra*** có các cấu trúc tương đương với phát biểu FORK và JOIN là cấu trúc PARBegin và PAREnd (hoặc COBegin và COEnd). Giả thiết rằng các tiến trình $S_1, S_2, ..., S_n$ có thể thao tác song song theo tiêu chuẩn Beinstein thì cấu trúc lệnh sau sẽ khởi động song song các tiến trình đó.

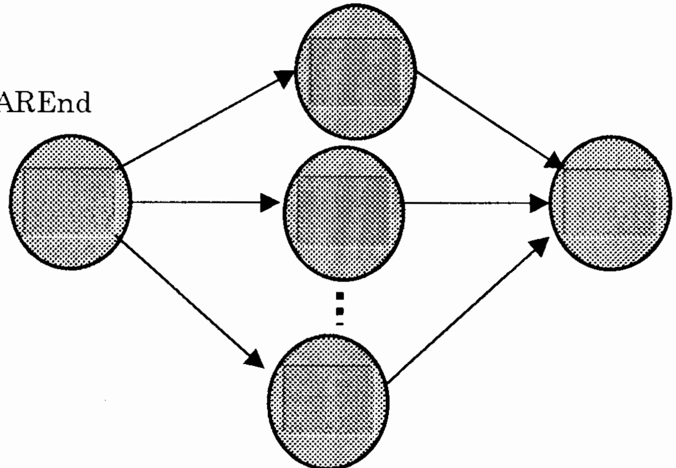
Begin

So;

PARBegin S1;S2;...;Sn;PAREnd

Sn + 1

End



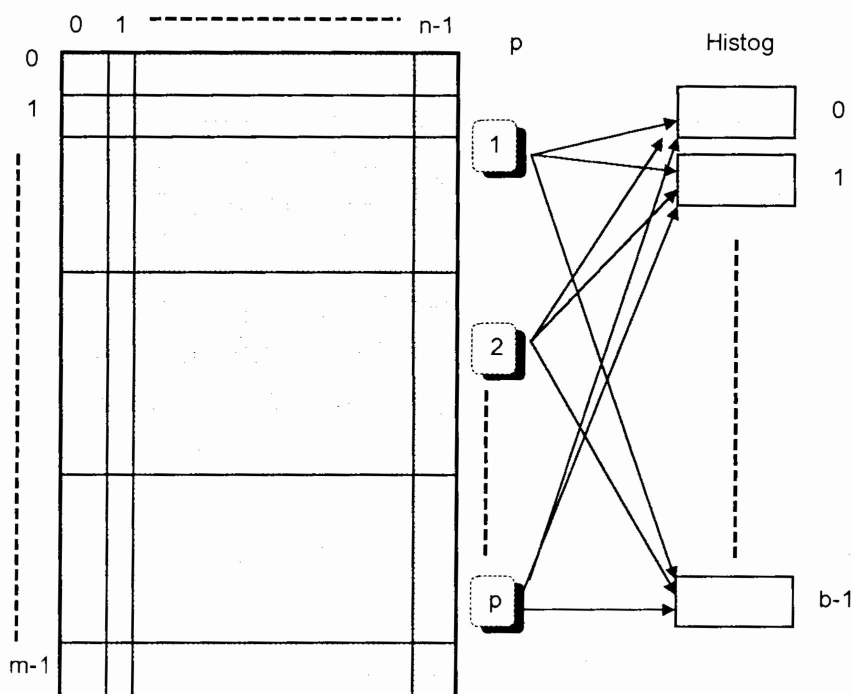
Hình 12.12. Các tiến trình song song s1, s2, ..., sn

Chương trình phải được viết sao cho dễ biên dịch, dễ kiểm tra mức liên kết để tổ chức các tiến trình song song và tổ hợp biến phục vụ cho các tiến trình đó.

PARBegin khai báo rằng chương trình hiện hành chia thành các thành phần con có thể hoạt động đồng thời. Điều này cho phép sử dụng các biến cục bộ và chia sẻ các biến toàn cục mà không gây xung đột trong hệ. Trên sơ đồ hình 12.12, PARBegin và PAREnd chỉ hoạt động sau khi So đã vào hoạt động. Sn+1 được khởi đầu chỉ khi tập hợp S1, S2,..., Sn đã kết thúc. Trong cấu trúc này, cần lưu ý một điều là một biến Vi bị thay đổi nội dung bởi thao tác Si thì biến đó không thể bị qui chiếu bởi thao tác Sj (với j<>i).

12.5. XỬ LÝ SONG SONG CÁC THAM SỐ ẢNH

Để minh hoạ khả năng cho hệ xử lý được thiết lập trên, chúng ta xét bài toán xử lý tham số Historgram cho ảnh đen trắng. Mỗi ảnh được biểu diễn bằng tập hợp các điểm ảnh - còn gọi là pixel. Sau khi đã xử lý được tham số này sẽ mở rộng cho các tham số khác và cho ảnh màu ở mọi dạng thể hiện



Hình 12.13. Xử lý p tiến trình chạy song song

Mỗi pixel có giá trị từ 0 đến (b – 1) tương ứng với một giá trị của thang đo độ xám của ảnh đen trắng. Nếu sử dụng một byte để chứa giá trị đó thì mỗi pixel sẽ có giá trị từ 0 đến 255. Historgram là một dạng tham số quan trọng vì nó chỉ ra tần xuất xuất hiện của mỗi giá trị trong thang đo độ xám của ảnh. Như vậy, đại lượng b phải được cập nhật và lưu trữ. Nếu cho Histog [O:b – 1] dùng để biểu diễn mảng

chứa số đếm tích luỹ của các giá trị độ xám 0,1,...,b-1 thì bức ảnh chữ nhật sẽ được biểu diễn bởi mảng hai chiều của các pixel [0: m-1,0: n-1] với m là dòng và n là cột.

Nếu pixel[i,j] biểu diễn giá trị thang đo độ xám tại toạ độ (i,j) thì dễ dàng xây dựng chương trình cho hệ đơn xử lý điều khiển việc cập nhật các giá trị:

```
Var pixel [0: m-1,0: n-1];
Var histog[0:b-1] :integer;
INITIAL histog[0:b-1]=0;  {Khởi tạo bộ đếm tần suất}
For i ←0 until  m-1 do
  For j ←0 until  n-1 do
            Histog [pixel [i,j]]---histog [pixel [i,j]] +1;
```

Thời gian cập nhật sẽ phụ thuộc vào tích m*n. Nếu m và n rất lớn thì chỉ riêng việc tính toán Histogram đã chiếm rất nhiều thời gian. Hơn nữa, nếu luồng thông tin chứa dữ liệu về ảnh truyền với tốc độ cao tới hệ xử lý thì nhiều khi hệ không kịp gia công và tính toán.

Bây giờ nếu hệ xử lý có p CPU ở chế độ chạy song song như cấu trúc hình 12.10 đã chỉ ra, thì ta có thể chia các dòng của ảnh thành p segment sao cho m/p = s dòng. Mỗi CPU phụ trách tính toán một mảng có s dòng và n cột. Như vậy chúng ta đã tạo ra p tiến trình độc lập có thể phát động song song. Tuy nhiên cần lưu ý là các tiến trình thao tác để hình thành Historgram phải chia sẻ mảng histog[0: b-1] để cập nhật nội dung cho các bộ đếm tần suất. Mức phân rã được tính là p (hình 12.13).

Sử dụng cấu trúc PARFOR ta có chương trình:

```
Var Histog[0:b-1] : integer : Shared { Khai báo biến chia sẻ }
Initial histog[0: b-1] =0                { Khởi đầu cho bộ đếm tần suất }
PARFor i  ← 1 until p do
 Begin
  Var pixel [(i-1) s: Si-1,0:n-1] : pixel;
   For k ←(i-1)*s until S*i-1 do
    For j ← 0 until  (n-1) do
        Csect histog [pixel [k,j]] do
          histog[pixel[k,j]] ←  histog[pixel[k,j]+1];
      end;
```

Thời gian cập nhật sẽ phụ thuộc vào tích s*n. Rõ ràng là tốc độ tăng lên rất nhiều. Tuy nhiên trong trường hợp này tốc độ không thể đạt p lần do các tiến trình phải chia xẻ bộ đếm histog[0:b-1] nằm ở ngoài bộ nhớ trung tâm nên phải xắp hàng

khi truy nhập bộ nhớ. Nếu dùng kiến trúc hệ có Cache thì hiệu quả sẽ cao hơn nữa vì việc truy nhập bộ nhớ chỉ xẩy ra vào giai đoạn cuối. Lúc này chương trình có thể thay bằng cấu trúc mới:

```
Var histog[0:b-1] : integer ;
Initial histog[0:b-1] = 0 ;
Var Ihistog[1: ;0:b-1] : integer;      { Biến cục bộ }
Initial Ihistog[1:p,0:b-1] = 0 ;
PARFor i < --- 1 until p do
  Begin
   Var pixel [(i-1)*s:S*i-1,0:n-1] : pixel;
    Var Ihistog[i,0:b -1] : integer;
    For k <--- (i-1)*s until S*i -1 do
       For j  <--- 0 until n-1 do
         Ihistog[i,pixel [k,j]] --- Ihistog [i,pixel [k,j]]+1;
  End;
   For j <--- 0 until b-1 do                { Tổng các histog thành phần }
    For i <--- 1 until p do
      Histog[i] <--- Histog[i] + Ihistog [i,j];
```

Kết luận: Xử lý song song cho phép thực hiện chương trình nhanh hơn và tận dụng tốt hơn tài nguyên trong hệ xử lý. Một tiến trình thường bao gồm nhiều bước tính toán và xử lý. Sử dụng phương pháp phận rã chức năng là phương pháp chính để phân chia nhiệm vụ thành các tiến trình con để tiến hành các thao tác tính toán song song Thuật toán xử lý ảnh Histogzam là một minh hoạ cho phép tăng tốc độ thao tác, đặc biệt khi dữ liệu ảnh là dữ liệuđa kênh từ vệ tinh truyền xuống.

Phụ lục

Phụ lục A. Bảng mã ASCII

Số TT	Mã hexa	Ký tự	Số TT	Mã hexa	Ký tự
0	00	NULL	31	1F	US
1	01	SOH	32	20	SP
2	02	STX	33	21	!
3	03	ETX	34	22	"
4	04	EOT	35	23	#
5	05	ENQ	36	24	$
6	06	ACK	37	25	%
7	07	BEL	38	26	&
8	08	BS	39	27	'
9	09	HT	40	28	(
10	0A	LF	41	29	)
11	0B	VT	42	2A	*
12	0C	FF	43	2B	+
13	0D	CR	44	2C	,
14	0E	SO	45	2D	-
15	0F	SI	46	2E	.
16	10	DLE	47	2F	/
17	11	DC1 (X-ON)	48	30	0
18	12	DC2 (TAPE)	49	31	1
19	13	DC3 (X-OFF)	50	32	2
20	14	DC4	51	33	3
21	15	NAK	52	34	4
22	16	SYN	53	35	5
23	17	ETB	54	36	6
24	18	CAN	55	37	7
25	19	EM	56	38	8
26	1A	SUB	57	39	9
27	1B	ESC	58	3A	:
28	1C	FS	59	3B	;
29	1D	GS	60	3C	<
30	1E	RS	61	3D	=

Bảng mã Ascii (tiếp)

Số TT	Mã hexa	Ký tự	Số TT	Mã hexa	Ký tự
62	3E	>	93	5D	]
63	3F	?	94	5E	^
64	40	@	95	5F	-
65	41	A	96	60	,
66	42	B	97	61	a
67	43	C	98	62	b
68	44	D	99	63	c
69	45	E	100	64	d
70	46	F	101	65	e
71	47	G	102	66	f
72	48	H	103	67	g
73	49	I	104	68	h
74	4A	J	105	69	i
75	4B	K	106	6A	j
76	4C	L	107	6B	k
77	4D	M	108	6C	l
78	4E	N	109	6D	m
79	4F	O	110	6E	n
80	50	P	111	6F	o
81	51	Q	112	70	p
82	52	R	113	71	q
83	53	S	114	72	r
84	54	T	115	73	s
85	55	U	116	74	t
86	56	V	117	75	u
87	57	W	118	76	v
88	58	X	119	77	w
89	59	Y	120	78	x
90	5A	Z	121	79	y
91	5B	[	122	7A	z
92	5C	\	123	7B	{
124	7C	|	126	7E	~
125	7D	}	127	7F	^

Phụ lục B. **Bảng mã điều khiển ascii**

Số TT	Mã hexa	Ký tự	Chức năng
0	00	NULL ^@	null (end string) - hết xâu ký tự
1	01	SOH ^A	start of heading- bắt đầu phần tiêu đề
2	02	STX ^B	start of text- bắt đầu phần văn bản
3	03	ETX ^C	end of text- kết thúc phần văn bản
4	04	EOT ^D	end of transmission- kết thúc truyền
5	05	ENQ ^E	enquiry- hỏi
6	06	ACK ^F	acknowlege- nhận biết
7	07	BEL ^G	bell- chuông
8	08	BS ^H	back space- lùi để xoá ký tự
9	09	HT ^I	tab horizontal tab dóng cột trên hàng
10	0A	LF ^J	line feed xuống dòng
11	0B	VT ^K	vertical tab dóng cội
12	0C	FF ^L	form feed sang trang
13	0D	CR ^M	carriage return về đầu dòng
14	0E	SO ^N	shift out dịch ra
15	0F	SI ^O	shift in dịch vào
16	10	DLE ^P	data line escape thoát dòng dữ liệu
17	11	DC1 (x-on)	^Q device controller1- TB đ/kh1
18	12	DC2 (tape)	^R device controller2- TB đ/kh2
19	13	DC3 (x-off)	^S device controller3- TB đ/kh3
20	14	DC4	^T device controller4- TB đ/kh4
21	15	NAK ^U	negative acknowlege- phủ nhận
22	16	SYN ^V	syncronous idle -đồng bộ
23	17	ETB ^W	end transmitte block- hết khối truyền
24	18	CAN ^X	cancel- huỷ
25	19	EM ^Y	end of medium hết không gian chứa
26	1A	SUB ^Z	substitute thay thế
27	1B	ESC ^[	escape- thoát
28	1C	FS ^/	file separator dấu cách tập tin
29	1D	GS ^]	group separator dấu cách nhóm
30	1E	RS ^^	record separator dấu cách bản ghi
31	1F	US ^-	unit separator dấu cách khối tin

MỤC LỤC

PGS, TS. ĐỖ XUÂN TIẾN

KỸ THUẬT VI XỬ LÝ
VÀ
LẬP TRÌNH ASSEMBLY CHO HỆ VI XỬ LÝ

Chịu trách nhiệm xuất bản : PGS, TS. TÔ ĐĂNG HẢI
Biên tập : ĐỖ THỊ CẢNH
Sửa bản in thử : LÊ MINH
Trình bày và chế bản : PHÒNG MÁY TÍNH
Vẽ bìa : HƯƠNG LAN

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
70 TRẦN HƯNG ĐẠO – HÀ NỘI

In 1000 cuốn, khổ 19 x 27 cm tại Xí nghiệp in NXB Lý luận chính trị
Giấy phép xuất bản số: 150-192, do cục xuất bản cấp ngày 4/2/2005
In xong và nộp lưu chiểu tháng 1 năm 2006.